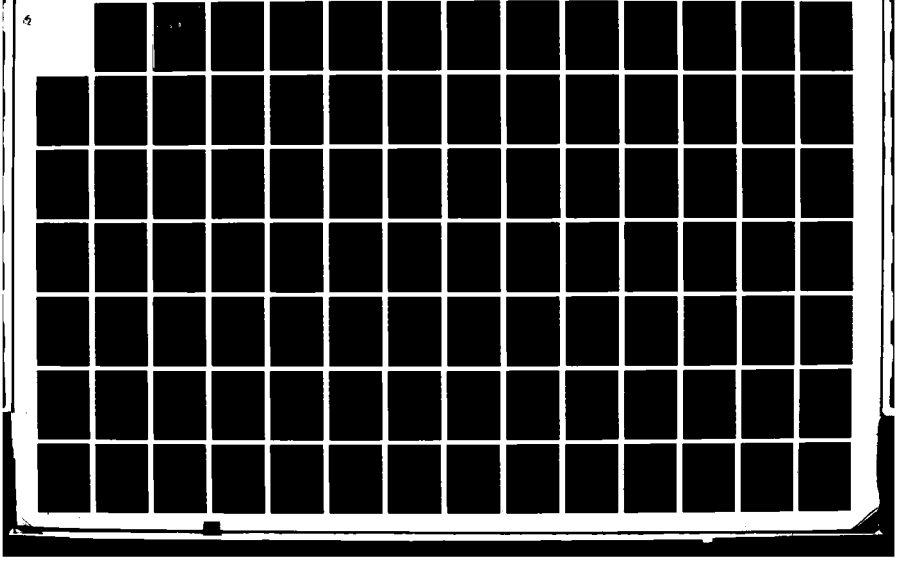
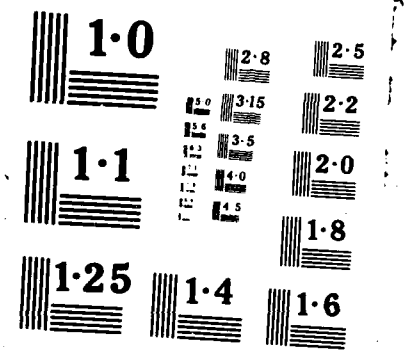
AD-A188 699 TASK ALLOCATION FOR EFFICIENT PERFORMANCE OF A DECENTRALIZED ORGANIZATION(U) MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION AND D.. C LEE

1/2

UNCLASSIFIED SEP 87 LIDS-TH-1706 N00014-85-K-0519 F/G 5/3 NL

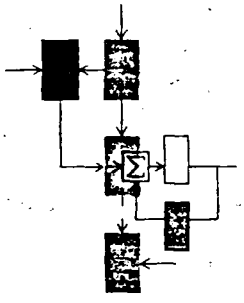



1·0
2·8
2·5
3·15
2·2
3·5
1·1
2·0
4·0
4·5
1·8
1·25
1·4
1·6

SEPTEMBER 1987

LIDS-TH-1706

Research Supported By:
N00014-84-K-0519
Office of Naval Research
Grant N00014-85-K-0519
(NR 649-003)



TASK ALLOCATION FOR EFFICIENT PERFORMANCE OF A DECENTRALIZED ORGANIZATION

Chonghwan Lee

Laboratory for Information and Decision Systems
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MASSACHUSETTS 02139

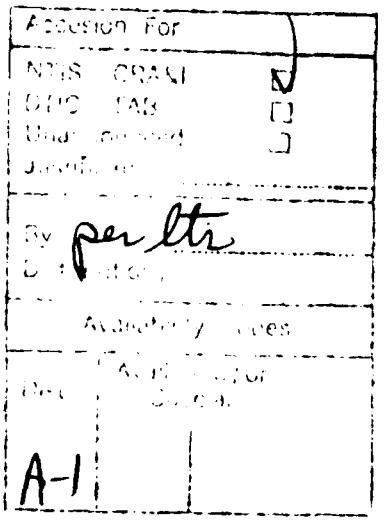
September 1987 LIDS-TH-1706

TASK ALLOCATION FOR EFFICIENT PERFORMANCE OF A DECENTRALIZED ORGANIZATION

by

Chonghwan Lee

This report is based on the unaltered thesis of Chonghwan Lee, submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the Requirements for the Degree of Master of Science in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology in August 1987. This research was supported by the Office of Naval Research under Grant N00014-85-K-0519 (NR 649-003).



Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

TASK ALLOCATION FOR EFFICIENT PERFORMANCE OF A DECENTRALIZED ORGANIZATION

by

CHONGHWAN LEE

B.S., Electrical Engineering and Mathematics
University of Maryland, College Park
(1985)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the Requirements for the Degree of

Master of Science
in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

August 1987

Signature of Author _____

Department of Electrical Engineering and Computer Science
August 7, 1987

Certified by _____

John N. Tsitsiklis
Associate Professor, Electrical Engineering
Thesis Supervisor

Accepted by _____

Professor Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

TASK ALLOCATION FOR EFFICIENT PERFORMANCE

OF A DECENTRALIZED ORGANIZATION

by

CHONGHWAN LEE

Submitted to the Department of Electrical Engineering and Computer Science on August 7, 1987 in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering and Computer Science

ABSTRACT

Task allocation scheme in an organization is discussed. The behavior of an organization is mathematically modeled by a decentralized gradient–like algorithm for additive cost functions. The objectives of allocation are reduction of individual load, speedy performance, and organizational security. The allocation scheme is sought for three types of organizations classified by the flexibility of their communication structure; namely, fixed organization, flexible organization, and semi–flexible organization.

Thesis Supervisor: Dr. John N. Tsitsiklis

Title: Associate Professor of Electrical Engineering

ACKNOWLEDGEMENTS

I would like to thank my research supervisor, John Tsitsiklis for his insightful guidance, generosity, and kindness. I also wish to thank my academic advisor, Michael Athans for his experienced advice in the course of my degree program.

I am grateful to my office mates in the Laboratory of Information and Decision Systems for their friendship. Especially, I thank Thomas Richardson and Zhi-Quan Luo for their encouragement and academic discussions.

Finally, I wish to express my thanks to my parents and sisters for their spiritual, emotional support in my life.

This research was supported by the Office of Naval Research under grant N00014-85-K-0519, (NR 649-003).

For God so loved the world that He gave His only begotten Son, that whoever believes in Him should not perish but have everlasting life. (*John 3:16*)

TABLE OF CONTENTS

Chapter 1. INTRODUCTION

1.1 Background

In an organization, how to partition a task and assign subtasks to divisions is an important problem. This topic is closely related to the issue of the 'organizational structure' and the 'efficiency of organizational performance'. An 'organization' could be a human organization, interconnected machines, or computer network. We will use the word 'agent' or 'division' in order to refer to a constituent of an organization – for example each person, machine, or a processor of a computer network. Here, what we mean by organizational structure is purely a communicational structure; namely, who should communicate with whom. The hierarchy among agents is not discussed. Efficiency generally means how quickly an organization can perform its task. In order to understand the issue clearly, let us consider the examples in Figure 1-1, 1-2, 1-3.

In the diagrams in these figures, the rectangles represent organizational tasks, and the circles represent agents of the organization. In Figure 1-1, dotted lines connecting task 1 and task 2 represent coupling between two tasks. This means that the decision making for task 1 affects the outcome of task 2 and/or vice versa. If tasks 1,2,3 are assigned to agents 1,2,3, respectively, agents 1 and 2 must communicate. Couplings among the divisional tasks necessitate certain communication links among the agents; therefore, they require a certain type of organizational structure. On the other hand, if the organizational structure is fixed, a task must be partitioned in such a way that subtasks assigned to disconnected agents are decoupled. For example, if we have a fixed organization like Figure 1-2, subtasks assigned to agents 1,2,3 must

be decoupled from those assigned to agent, 4,5.

As for the efficiency of organizational performance, consider Figure 1-3. Suppose we have three agents with equal capacity and a global task comprised of 6 mutually decoupled, equal-size subtasks. Intuitively, we know that (b) is a more efficient allocation than (c), because with partition (b) the task will be executed sooner than with partition (c).

One of the obstacles to the discussion of this problem of task partition is the conceptual difficulty of constructing mathematical models which precisely describe the behavior of organizations. A particular mathematical model has been developed in [1] based on a decentralized gradient-like algorithm. This algorithm is basically a descent–type optimization algorithm for parallel processing. Each processor has a local cost function to optimize, and each processor has its own variable that it updates at each iteration. Each local cost function may depend, however, on the variables to be updated by other divisions. With each processor updating its own variable and communicating information with other processors, the algorithm achieves the goal of optimizing the sum of local cost functions. Such a decentralized gradient– like algorithm for additive cost functions could describe the behavior of an organization of boundedly rational agents, as they adjust their decisions toward the objective of minimizing an organizational cost function. The minimization of the cost function is viewed as the organizational task, and the value of each variable in the cost function represents a decision made by the associated division. Alternatively, the value of each variable can be viewed as the mode of operation of the associated division if we set the organizational objective to be the optimal operation, in the sense that the variables have the values minimizing the cost function. An iterative minimiza-

tion process models the boundedly rational agents, who make tentative decisions based upon available information of the decisions made in other divisions and adjust decisions as they gain additional information. Partitioning the organizational task mathematically translates to decomposing the global cost function into the sum of subcost (or local cost) functions. The problem of partitioning an organizational task will be formulated in this mathematical framework in the next section.

1.2 Problem statement

It is useful to specify the purposes of partitioning the task, in order to formulate the problem in a way that is relevant to some real world situations or applications. The purposes of the partition are classified as follows:

1. Reduction of individual load:

When the task is complex or the scale of the task is big, a single agent with limited capacity simply cannot handle it. The agent's memory is limited. Its power is limited. Actually, this is the reason why organizations are formed. We can reduce the load of an agent by partitioning the task. In our formulation, the load of each agent will be measured by the complexity of the subcost function assigned to that agent. (The measure of the complexity of the subcost function will be precisely defined later.)

2. Speed of performance:

Even if one agent may be able to handle the whole task, it is often better to have the agents in the organization share the task so that they finish the task sooner. The speed of organizational performance may be represented by the speed of convergence

of the decentralized algorithm discussed earlier.

3. Security:

Under special conditions such as defense projects, it is not desirable to set the operation under the control of a single agent. The 'decentralized gradient-like algorithm for an additive cost function' is an appropriate model of the organization serving this purpose. In this algorithm, a processor does not know what other processors' subcost functions are. This means that each agent keeps only a fraction of organizational secrets. We also want to minimize the amount of communication because an organization can reduce the leakage of secrets by keeping the amount of communication minimal. The amount of communication will be defined in two different ways in order to model certain different situations.

Therefore, the task should be partitioned so that these purposes are fulfilled. For convenience of exposition, we assume that the global cost function is quadratic, and its minimization represents the organizational task. The only thing that matters is the coupling between variables introduced by the cost function. Thus, assuming a quadratic function is no loss of generality. We also assume that the complexity of a subcost function is measured by the number of terms in that subcost function. Mathematically, our problem is :

Given a quadratic cost function, find a decomposition into the sum of subcosts such that

Objective 1 :

The number of terms in each subcost is small.

Objective 2 :

The speed of convergence of the decentralized gradient-like algorithm is maximized.

Objective 3 :

The amount of required communication is minimized. (If the subcost assigned to agent i depends upon a decision variable determined by agent j, agent i and agent j are required to communicate with each other according to the model.)

It is conceivable that one may formulate a multi-objective optimization problem, which accounts for all of these three objectives. However, our understanding of the decentralized gradient-like algorithm is too restricted to solve it. In fact, analytical understanding of the speed of convergence seems impossible. Moreover, these three objectives may well be conflicting with one another. Therefore, we have to deal with each of the three objectives separately. The following are some feasible problems.

Given a quadratic cost function, find a decomposition into the sum of subcost functions that

Problem 1 :

minimizes the number of terms of the agent with highest load; under the constraint that the amount of required communication is less than a certain number or under a constraint that the communication structure of an organization is restricted.

Problem 2 :

minimizes the amount of required communication; under the constraint that the number of terms (representing load) assigned to each processor is less than a certain

integer.

Problem 3 :

maximizes the speed of convergence.

When designing an organization with a severe constraint on the communication structure, a designer needs to compute the maximum load of divisions in order to figure out the required capabilities of divisions. Problem 1 provides a mathematical framework to study this issue. Problem 2 deals with the situation where the capabilities of each division are already fixed. The issue is how to allocate the tasks cleverly in order to minimize the amount of communication. Solutions to Problem 2 can also be used as a measure of coupling of the task. It shows how much cooperation it takes to perform the task when information is distributed.

More detailed, rigorous formulation of these problems will be presented in Chapter 2.

1.3 Literature Survey

There has been little literature concerning mathematical formulation of organizational behaviors, especially task assignment strategy. Reference [1] suggests using decentralized descent-type algorithms as a model of behavior of boundedly rational human decision makers. The mathematical framework of [1] will be adopted for our research. Reference [2] discusses a design method for certain classes of human organizations. It does not develop a specific mathematical model for human organizations. Rather, it suggests a general design method that applies to some

human organizations that already have tractable analytical model. Other literature mostly models an organization in an information theoretic framework. Reference [3] develops a model for interacting decision makers. A decision maker is modeled by a processor that takes input (stimulus), processes it, and produces an appropriate output(response). Each processor has several algorithms to choose for this information processing. Internal decision strategies of choosing an algorithm are introduced. Total activity of a processor and the performance measure are expressed in terms of these internal decision strategies, and bounded rationality is modeled by the constraints on total activity of a processor. Reference [3] characterizes the internal decision strategies that give optimal or satisficing performance under constrained or unconstrained total activity. In [4] each member of an organization is again modeled as an information processor. Techniques for allocating information processing tasks to members are discussed. Creating self–contained tasks is analogous to decomposing a global cost function in our formulation. Self–contained tasks translate to decoupled subcost functions in our formulation. Reference [5] uses a queueing network model to describe a team of two decision makers. Two decision makers are modeled as two service stations with different processing capabilities, different processing rates (expertise), and common information. There are three classes of tasks, and tasks arrive at the team of decision makers dynamically. Reference [5] discusses the optimal policy to select/allocate these tasks.

Our proposed, mathematical model of an organization is simpler than the information processor models of [3], [4], [5] in the sense that a 'Decentralized-descent algorithm' does not describe an organization's interaction with the external environment. On the other hand, our model is more explicit in showing interactions between divisions or the coupling of organizational task.

The mapping problem in [8] is quite similar to some of our proposed problems. Consider a program made up of several modules. When these modules are executed in parallel, some modules must communicate with each other. When parallel processors are incompletely connected, a pair of modules that must communicate with each other should be executed by neighboring processors. The mapping problem is how to assign modules to processors so that the number of such successfully placed pairs of modules are maximized. A set of modules and their communication requirement are analogous to our subtasks and their couplings. However, there are a few differences. In the mapping problem, communication structure of modules and the interconnection of processors are fixed. In our problem, couplings among subtasks depend upon how we decompose a global task. How we decompose a global task is a more important issue than how we map subtasks to agents.

1.4 Outline

In Chapter 2, the mathematical model recruited for the discussion will be explained in detail. Mathematical realization of suggested problems will be made in various ways. For each realization, a strategy for solution will be briefly indicated. In subsequent chapters (Chapter 3 – Chapter 4), each formulation will be discussed in detail, and solutions for them are discussed in detail.

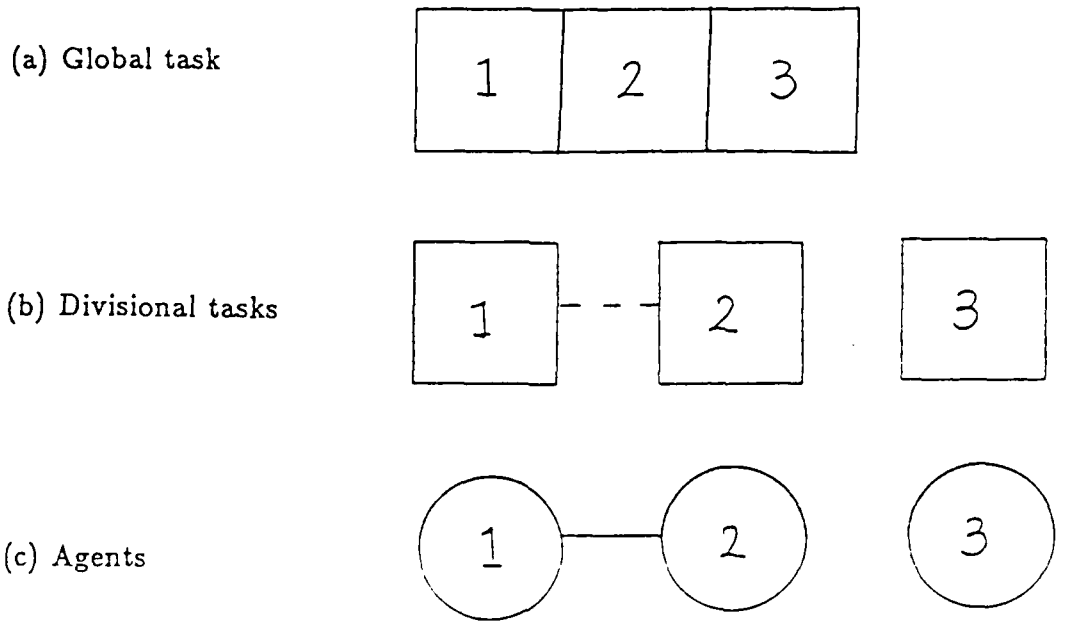


Figure 1-1

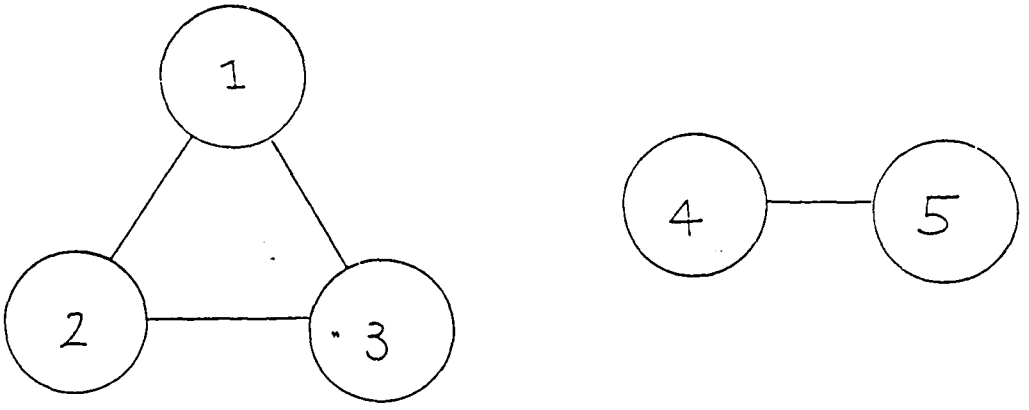
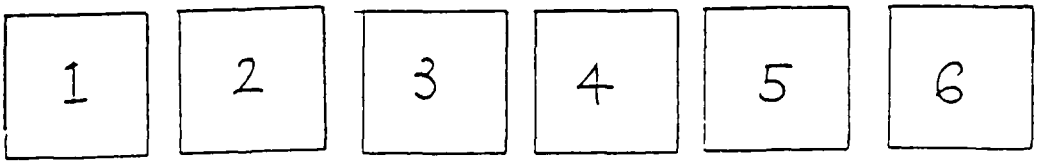
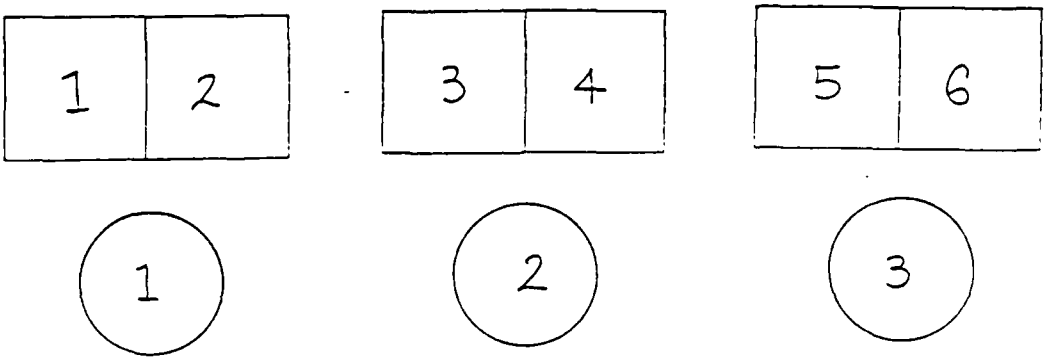


Figure 1-2 Organizational structure

(a) Global task



(b) Assignment



(c) Assignment

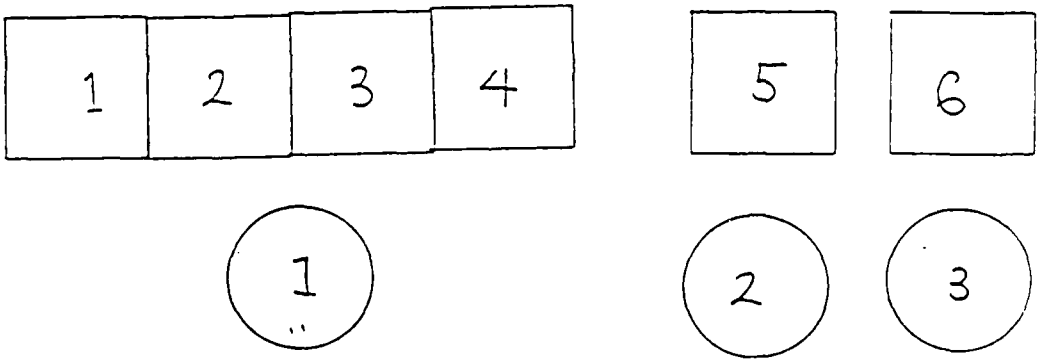


Figure 1-3 Comparison of two assignments

Chapter 2 . MATHEMATICAL MODEL

2.1 Decentralized Gradient–like Algorithm for an Additive Cost Function

In this section we introduce the decentralized gradient–like algorithm for an additive cost function and make some important observations. We also discuss how this algorithm models the behavior of an organization.

For the cost function, $J(x) = J(x_1, x_2, ..., x_M) = \sum_{i=1}^{M} J^i(x_1, x_2, ..., x_M)$, the processor i has the subcost J^i and is responsible for the variable x_i. The algorithm is summarized as follows:

Algorithm 2.1.1

At each iteration n,

1. Each processor j evaluates the partial derivative $\lambda_i^j(n) = \frac{\partial J^j}{\partial x_i}(n)$ for every i such that J^j depends on x_i.
2. Each processor j transmits $\lambda_i^j(n)$ to processor i, for every processor i such that J^j depends on x_i.
3. Each processor i updates x_i according to
   $$x_i(n+1) = x_i(n) - \gamma_i\left(\sum_{j=1}^{M} \lambda_i^j(n)\right)$$
   (Here, γ_i is a positive scalar stepsize.)
4. Each processor i transmits $x_i(n+1)$ to all processors j that depend upon x_i.

Algorithm 2.1.1 is a synchronous version of a decentralized gradient-like algorithm. Since processors communicate all the necessary information at the end of each iteration, this algorithm is mathematically doing the following:

$$\tilde{x}(n+1) = \tilde{x}(n) - D\nabla J(\tilde{x}(n))$$

where $D = diag(\gamma_1, \gamma_2, ..., \gamma_M)$.

This is a steepest descent algorithm with scaling, and γ_i is a scaling factor of the i-th component of the gradient.

An asynchronous version of Algorithm 2.1.1 is basically the same as Algorithm 2.1.1 except that the asynchronous version does not require x's and λ's to be transmitted at each iteration. It also allows communication delay. It has been shown in [1] that the asynchronous version of Algorithm 2.1.1 converges if the λ's and x's are transmitted frequently enough. A precise mathematical statement is in [1].

An important observation is that for all the pairs i and j such that J^j depends upon x_i, x_i must be transmitted from processor i to j, and λ_i^j must be transmitted from processor j to i. It is not necessary that all the processors communicate with each other. Therefore, a path is only required between certain pairs of processors. A set of subcost functions requires a certain class of communication networks.

We view minimizing the global cost function, $J(x_1, ..., x_M) = \sum_i J^i$ as a task of an organization. Values of M variables, x_1,x_2,....,x_M represent decisions the organization has to make to accomplish the task. In order to reduce the work load and to enhance the security of organizational information, an organization distributes the authorities to make such decisions to M divisions. Each division, i has an authority to determine the value of x_i. Each subcost J^i specifies the subtask entrusted to division i. J^i is known only to division i. Therefore, each division does not know what others are doing; this is to serve the security objective. Each state of

computation shows how far the organization has advanced in performing the task. Iterative computation models boundedly rational divisions (or human decision makers), which make decisions based upon partial knowledge and update the decisions as they acquire more information, hopefully in a direction that decreases costs.

2.2 Problem formulation

In this section we rigorously formulate the proposed problems within the mathematical framework described in section 2.1. In order to make our discussion concise, we define the following symbols in advance.

$J(x_1, x_2, ..., x_M)$: the global cost function

DM_i : decision maker, processor, agent, or division which is responsible for the value of x_i

J^i : a subcost function

(If a certain term of the global cost function is in J^i, we say that the term is 'assigned to DM_i'.)

$nt(i)$: the number of cross terms of J^i

(We assume that J^i's are quadratic; therefore, we can count the number of cross terms. We count a product of two variables as one term. For example, for $J^1 = x_1^2 + x_2(x_3 + x_4)$, $nt(i) = 2$.)

λ_i^j : a partial derivative $\frac{\partial J^j}{\partial x_i}$

$d(DM_i)$: degree, the number of links incident upon DM_i

$L(G)$: a set of nodes of a graph G whose degree is 1. (leaves)

$max_i\ nt(i)$: maximum of $nt(i)$ over all i.

$G_Q = (V_Q, E_Q)$: undirected graph with a node set V_Q and an edge set E_Q

$G = (V, E)$: undirected graph with a node set V and an edge set E_Q

Given a cost function $J(x_1, x_2, ..., x_M)$, there are many ways to decompose it as the sum of M subcost functions, J^i's such that $J(x_1, x_2, ..., x_M) = \sum_{i=1}^{M} J^i$. Basically, our problem is to find a decomposition that meets the specified objectives.

2.2.1 Global cost function

As mentioned ahead, the study is restricted to the quadratic cost function,

$$J(x_1, ...x_M) = (x_1, ..., x_M)Q(x_1, ..., x_M)^T$$

where Q is a symmetric matrix. In order to represent J, we will often use an undirected graph $G_Q = (V_Q, E_Q)$, where $V_Q = \{1, 2, ..., M\}$ and $E_Q = \{(i,j)|Q(i,j) \neq 0\}$. ($Q(i,j)$ is an entry of Q, i-th row and j-th column.) Conceptually, the matrix Q or the graph G_Q characterizes an organizational task. For example, the dimension of Q represents the scale of the organizational task. Dense Q represents a situation where the organizational task is very intricate in nature. In other words, a decision made by a division affects many other divisions, no matter how the task is partitioned. Therefore, sophisticated coordination or cooperation [7] strategy is desired for this kind of task. On the other hand, a diagonal matrix, Q represents a situation where each division can perform its own task without any interaction with other divisions.

In our formulation we assume that G_Q is connected. For G_Q not connected, one can always permute the rows and columns of Q and transform it to a block–diagonal matrix, Q'. Therefore, if Q' has k diagonal blocks, the cost J can be decomposed into k subcosts completely decoupled, and the problem is reduced to our formulation by considering each connected piece of G_Q separately. We also assume that the diagonal elements of Q are non–zero.

2.2.2 Assumptions on decomposition

We assumed in the previous section that

$$Q(i,i) \neq 0, \quad i = 1, 2, ..., M$$

This means that the global cost function J does not miss any square term. We now mandate that each square term $Q(i,i)x_i^2$ must be in J^i in the decomposition. The interpretation of this assumption is that each division (DM_i) must take over the partial task (x_i^2) that does not involve decisions of other divisions. This assumption makes mathematical formulation clear.

If J^j depends upon x_i, there must be a path from DM_i to DM_j through which x_i will be communicated, and a path from DM_j to DM_i through which λ_i^j will be communicated. This fact was mentioned in the previous section. We will consider the issue of decomposing a global cost function under two different assumptions concerning the communication between such pairs of processors.

The first assumption is that if J^j depends upon x_i, there must be a direct bidirectional link between DM_i and DM_j. We call this assumption '**Direct Communication**'. This assumption is motivated by the security purpose. If information

of x_i or λ_i^j travels from a source to destination through other DM's, other DM's have chances to acquire that information. This is not desirable for the security of information, so we want a direct link between DM_i and DM_j. This assumption is also motivated by the speed consideration. If information is relayed by intermediate DM's as described above, delays of information transfer will be longer, and intermediate DM's will be unnecessarily overloaded by carrying information in transit.

The second assumption is more relaxed. We only require that DM_i and DM_j be connected through a series of links if J^i depends upon x_j. Information on the value of variables or partial derivatives can be communicated en route other processors between this kind of pair of processors. We call this assumption **'Indirect Communication'**.

From these assumptions, the following lemma immediately follows.

Lemma 2-1

Let the global cost function be $J = x^T Q x$. If $Q(i,j) \neq 0$, there must be a path between DM_i and DM_j in G no matter how J is decomposed. Moreover, DM_i and DM_j are within two hops under 'Direct Communication' assumption.

Proof

Let us say that $x_i x_j$ is in J^k in a decomposition. Since J^k depends upon x_i and x_j, the value of x_i and the partial derivative λ_i^k must flow between DM_i and DM_k. Therefore, a path between DM_i and DM_k is required. Likewise, a path between DM_j and DM_k is required. Therefore, there must be a path between DM_i and DM_j in G.

Under the 'Direct Communication' assumption a link must exist between DM_i and DM_k, and between DM_j and DM_k. Therefore, DM_i and DM_j are within two hops in G. (If $k = i$ or $k = j$, the result still holds, trivially.)

2.2.3 Measure of objectives

In this section the mathematical definitions of three objectives of decomposition are discussed; namely,

Objective 1: balance of loads

Objective 2: the amount of communication

Objective 3: the speed of convergence

Objective 1 reflected the idea that one of the purposes of a decomposition is to balance loads that fall on each division of an organization. In our model (Decentralized gradient-like algorithm for additive cost functions), the load on each division DM_i arises from the task of minimizing a subcost function J^i. Therefore, a reasonable measure of the load on DM_i will be the amount of effort or resources DM_i has to exert to minimize J^i. This effort or resources include memory spaces and computational operations. In Algorithm 2.1.1, the local memory of processor DM_i must contain the the subcost function J^i, the set of variables,

$$SV(i) = \{x_j^i | J^i \text{ depends upon } x_j\},$$

the set of partial derivatives

$$SP(i) = \{\lambda_i^k | J^k \text{ depends upon } x_i\}.$$

Notice that

$$|SV(i)| = \textit{the number of variables } J^i \textit{ has}$$

$$|SP(i)| = \textit{the number of subcost functions that depend upon } x_i$$

As for the computational operations, DM_i must *receive* the values of partial derivatives from $|SP(i)|$ processors in order to update x_i. Also, DM_i must *add* $|SP(i)|$ partial derivatives in order to update x_i. (Recall $\frac{\partial J}{\partial x_i} = \sum_k \frac{\partial J^k}{\partial x_i}$.) DM_i must *send* the updated x_i to $|SP(i)|$ processors whose subcost function depends upon x_i. DM_i also has to *compute* $|SP(i)|$ partial derivatives and *send* to $|SP(i)|$ corresponding processors.

Therefore, the accurate measure of the load on processor DM_i will be a function of not only J^i but also the whole decomposition of J. The sum of the following will be an accurate measure of load on DM_i: the cost of memory space for storing J^i, the cost of storing variables proportional to $|SV(i)|$, the cost of storing partial derivatives proportional to $|SP(i)|$, the cost of receiving values of partial derivatives proportional to $|SP(i)|$, the cost of adding $|SP(i)|$ partial derivatives, the cost of transmitting its variable proportional to $|SP(i)|$, the cost of computing $|SP(i)|$ partial derivatives, the cost of transmitting these partial derivatives proportional to $|SP(i)|$.

In order to simplify our analysis, we will use an approximate measure of loads on each processor. We simply define the load of processor DM_i to be *the number of cross terms J^i contains* . This definition is possible because we are using the quadratic function as a global cost function in our model. We can count the number of cross terms in a quadratic cost function. The advantage of this definition is that we can view the decomposition process as assigning each cross term to processors. Every time a cross term in a global cost function is assigned to a processor, the load on this processor is increased by one. The analysis is simplified a lot in this way.

This definition of load is a relatively faithful approximation to the accurate measure of load stated above. If the number of terms grow in a subcost function, the required memory space to store this subcost function must also become big. Generally, if the number of cross terms in J^i is big, J^i depends upon many variables (big $|SV(i)|$). We use $nt(i)$ to denote the number of terms assigned on DM_i.

We define the measure of balance to be the maximum of these numbers of terms over all processors, i.e. $max_i\ nt(i)$. For the same global cost function J, smaller $max_i\ nt(i)$ means more balanced decomposition.

The amount of communication, again, will be extremely complicated if we want to follow our algorithmic model literally. We would have to count every single message transmitted and received. Theoretical analysis of this is impossible. Even if we were able to count all the message flows, the result would mean nothing more than the cost of communication in parallel processing. We present two approximated measure of the amount of communication, and explain how these measures can be interpreted more meaningfully.

First, we consider the number of processor pairs that need to communicate with each other. We have stated that DM_i and DM_j must communicate with each other if J^i depends upon x_j or J^j depends upon x_i. We count the number of such pairs and use this number as the amount of communication. As long as a pair of processors need to communicate with each other, we do not care how many bits of information need to be actually exchanged. This is a convenient simplification. Under the first assumption on decomposition in section 2.2.2 ('Direct Communication'), this number of processor pairs is the total number of necessary links in a set of processors in order

to perform the organizational task with the corresponding partition of tasks. We call this measure of the amount of communication TL (Total number of Links). This measure, TL can also be interpreted as the measure of risk of the leakage of organizational secret if every direct channel has the same probability of information leakage. If an organization has a fixed structure in terms of its communication ability (this will be explained in Section 2.2.4), TL is a total number of edges in the graph representing the fixed organizational structure.

As an alternative, we define the amount of communication slightly differently from TL. Decomposing a quadratic global cost function $J(x_1, x_2, ..., x_M)$ can be viewed as assigning each cross term of J to one of processors, $DM_1, DM_2, ..., DM_M$. If the cross term $x_i x_j$ is allocated to DM_k, $k \neq i$,$k \neq j$, two links must exist; namely, a direct link from DM_i to DM_k and a direct link from DM_j to DM_k. If the cross term $x_i x_j$ is allocated to DM_i or DM_j, one link must exist; namely a link between DM_i and DM_j. From this fact we can define the measure of necessary communication as the sum of the number of links introduced by this rule over all cross terms. We call this number 'RL' (Repeated number of Links). Obviously, this is not an exact number of necessary links under the assumptions of previous chapters. The reason is that the number of links can be counted repeatedly for the same pair of processors that need to communicate with each other. (For this reason RL will be often called **'superposed link'**.) However, this definition of communication load serves as an approximate measure. There is another interpretation for this measure RL. The work load of each division $nt(i)$ is defined to be the number of terms assigned to that division DM_i. Let us imagine that each division consists of individuals, and each individual is responsible for one term. In the same division, the decision variable of its own is known to every individual. However, for security purpose, individuals are

not allowed to inform one another with the values of decision variables they received from other division. In this set-up, division DM_i must communicate with all other individuals in other divisions who are assigned with a cross term involved with x_i. Therefore, RL is a total number of communication channels in this set-up.

As for the speed of convergence, the best measure for mathematical analysis would be the rate of convergence of the algorithm. However, the reasonable, coherent definition of the rate of convergence for distributed and asynchronous algorithm is very messy in terms of its mathematical expression. Also, it is very difficult to compute. This theoretical approach becomes mathematically intractable. For example, the speed of convergence is related to many factors other than how we decompose the global cost function. It depends upon the stepsize and the frequency of communication between processors as well. Finding analytical relations between these factors and the speed of convergence is mathematically intractable. The only way to gain some understanding is to perform insightful computer simulations. One would want to show through simulation that balanced assignments of subcosts in general improve the speed of convergence. Therefore, the whole analysis of the speed of convergence should inevitably be done heuristically. We suggest that one should simulate Algorithm 2.1.1 and count the number of iterations of the simulation program until it approaches the solution sufficiently close. We suggest using this quantity as a measure of the speed of convergence. We suggest one should design the simulation program in a way that this iteration count represents the real time taken to run Algorithm 2.1.1. Now, the issue is how to decompose a global cost function in order to meet this objective of fast convergence. It is intuitively clear that the balanced decomposition ends up with faster speed of convergence. Therefore, we

conclude that Objective 2 (fast convergence) of Section 1.2 is embedded in Objective 1 (balance of loads) in general.

2.2.4 Three types of organizations

We have rigorously stated the objectives of decomposition of a global cost function (or partitioning an organizational task) and our assumptions. In this section we classify organizations into three classes; namely, fixed organization, flexible organization, semi–flexible organization. This classification is based upon the flexibility of their communication structure. ('Communication structure', here, simply means which pairs of divisions or processors can communicate directly. It can be represented by a graph, where each node symbolizes a division or a processor, and each link symbolizes a pair of processors that can communicate directly with each other.) Our problem is to decompose the global cost function in order to meet the objectives (section 2.2.3) under the assumptions in Section 2.2. This problem shows distinctive lineaments when solicited for these different types of organizations.

In a fixed organization, the communication structure is strictly fixed. Each division is a priori determined to be responsible for a certain decision variables. (Decision variables of an organization mathematically translates to variables of J in section 2.2.1.) Which pairs of these divisions can directly communicate with each other is again determined a priori. Since the task allocator or a task allocating algorithm does not have an authority to specify or modify the communication structure of an organization, it has to find a decomposition that only requires communication through predetermined links. (Recall if J^i depends upon x_j, a communication between a processor responsible for x_i and one for x_j must communicate with each

other.)

In a flexible organization, the communication structure is not predetermined. The task allocator or task allocating algorithm has an authority to impose a communication structure of an organization.

In a semi–flexible organization, the communication structure of an organization is fixed. However, which processor is responsible for which decision variable is not determined a priori. The task allocator or the task allocation algorithm has an authority to determine the mapping from a set of decision variable to a set of processors as well as the decomposition of a global cost function.

2.2.5 Specific formulations

Our problem is to find a decomposition that meets the objectives specified in section 2.2.3 under the assumptions specified in section 2.2.1 and 2.2.2. We are concerned with two objectives; namely, the balance of loads and the reduction of the amount of communication. We have concluded that the speed of convergence (Objective 3) is embedded in the objective of the balanced load. We suggested two definitions for the amount of communication in section 2.2.3. In section 2.2.2, we suggested two different assumptions on decomposition; namely, 'Direct Communication' and 'Indirect Communication'. Since we have two definitions of the amount of communication to select and two assumptions to choose from, we can think of four combinations for the formulation of problems. We can match each of these four problems to three different types of organizations specified in section 2.2.4. Therefore, we have twelve cases to consider.

2.2.5-1 'Direct Communication'

< TL as the amount of communication >

A. Fixed organization

Since the organization has a fixed communication structure, the total number of links is also fixed. TL is invariant of a decomposition of J. Therefore, we can only consider the balance of load. Our problem is, then, to find a decomposition that minimizes $max_i\ nt(i)$ such that 'Direct Communication' assumption is satisfied.

Load Balancing in a Fixed Structure

Given $G_Q = (V_Q, E_Q)$ and a graph $G = (V, E)$, find a decomposition that minimizes $max_i nt(i)$.

B. Flexible organization

Let us consider TL as our objective. If we do not care about the other objective (the balance of loads), the problem of minimizing TL becomes trivial. If we assign the global cost function to a single processor, TL is $M - 1$, where M is the number of variable. ($M = |V_Q|$) TL is $M - 1$ because J has all the variables $x_1, x_2, ..., x_M$. $M - 1$ is a minimum TL because G has M nodes, and connected by Lemma 2-1. (Recall that we assumed G_Q is connected.)

If we care about the balance of load while minimizing TL, the problem becomes nontrivial.

Minimal Link

Given $J = x^T Q x$, nt_i^*, find a decomposition that minimizes TL such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

This problem models a situation where each division has a limited capacity (nt_i^*), and we want to partition an organizational task (minimizing J) such that the size of divisional task $(nt(i))$ is within the capacity of corresponding division. The aim of decomposition is to minimize necessary amount of communication (TL).

Now, let us consider $max_i\ nt(i)$ as our objective. Again, if we do not care about the amount of communication, the problem becomes trivial. We equally split the cross terms of J and assign pieces to processors. It will produce $max_i\ nt(i) = \lceil \frac{|G_Q|}{M} \rceil$ If we restrict the amount of communication, the problem becomes nontrivial.

Load Balancing with Limited Link

Given $J = x^T Q x$, TL^*, find a decomposition that minimizes $max_i\ nt(i)$ such that $TL \leq TL^*$.

C. Semi–flexible organization

Like Fixed organization, TL is fixed in this case. The problem is to find an assignment of decision variables and a subcost functions to processors that minimize $max_i\ nt(i)$.

Mapping for Load Balancing

Given G_Q and $G = (V, E)$, find a decomposition $J(\tilde{x}) = \sum_i J^i$ and a matching between $\{x_1, x_2, ..., x_M\}$ and V, so that these minimize $max_i\ nt(i)$.

Even before the issue of finding a decomposition $J(\tilde{x}) = \sum_i J^i$ is discussed,

finding a feasible matching between $\{x_1, x_2, ..., x_M\}$ and V is a notrivial problem. It will be discussed in detail in Chapter 3.

< RL as the amount of communication >

A. Fixed organization

Let us consider RL as an objective. The problem is, then,

Minimal Superposed Link in a Fixed Structure

Given a graph G, G_Q, and nt_i^*, find a decomposition that minimizes RL such that the message is only transferred through the edges of G, and such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

(We can cast off the constraint on the capacity of each processor by letting $nt(i) = \infty, \forall i$.)

We can also consider $max_i nt(i)$ as our objective.

Load Balancing with Limited Superposed Links in a Fixed Structure

Given a graph G, G_Q, and RL^*, find a decomposition that minimizes $max_i nt(i)$ such that the message is only transferred through the edges of G, and such that $RL \leq RL^*$.

B. Flexible organization

As in the case of (TL, Flexible organization), the problem of minimizing RL

becomes trivial if we do not care about the balance of loads. We know that RL is increased either by one or two whenever we assign a cross term. If we do not care about the balance of loads, for each cross term $x_i x_j$, we assign it to either DM_i or DM_j. This way, RL is increased only by one for each cross term. Therefore, RL is $|E_Q|$, and this is minimum. As in the case of (TL, Fixed organization), we can put constraints on the capacity of each processor.

Minimal Superposed Link

Given $J = x^T Q x$, nt_i^*, find a decomposition that minimizes RL such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

If we use $max_i nt(i)$ as our objective, the problem is

Load Balancing with Limited Superposed Link

Given $J = x^T Q x$, RL^*, find a decomposition that minimizes $max_i\ nt(i)$ such that $RL \leq RL^*$.

C. Semi–flexible organization

Mapping for Minimal Superposed Link

Given G_Q, $G = (V, E)$, and nt_i^* for $i = 1, 2, .., M$, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize RL such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

(We can cast off the constraint on the capacity of each processor by letting $nt(i) = \infty, \forall i$.)

Mapping for Load Balancing with Limited Superposed Link

Given G_Q, $G = (V, E)$, and RL^*, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize $max_i\ nt(i)$ such that $RL \leq RL^*$.

2.2.5-2 'Indirect Communication'

$< TL$ as the amount of communication $>$

A. Fixed organization

Since G is fixed, TL is also fixed. The only objective we can consider is $max_i nt(i)$. The problem is, then,

Given $G_Q = (V_Q, E_Q)$ and a graph $G = (V, E)$, find a decomposition that minimizes $max_i\ nt(i)$.

We have assumed that G_Q is connected. Therefore if $|V_Q| = |V|$, and G is not connected, feasible decomposition does not exist. On the other hand, as long as G is connected, the problem becomes trivial. Since we allow indirect communication, any pair of processors can communicate their messages with each other as long as G is connected. Therefore, we can equally split cross terms of J and assign pieces to processors. Then,

$$max_i\ nt(i) = \lceil \frac{|E_Q|}{M} \rceil$$

B. Flexible organization

In this case we can trivially find a decomposition that minimizes both TL and $max_i\ nt(i)$ at the same time. We have assumed that G_Q is connected, so G ends up being connected for any decomposition of J. Therefore, minimum TL is $M-1$. (G is a tree.) Moreover, since we allow indirect communication, any pair of processors can communicate with each other in a tree–structured organization. Thus we can achieve

$$max_i\ nt(i) = \lceil \frac{|E_Q|}{M} \rceil$$

C. Semi–flexible organization

TL is fixed. Allowing indirect communication, again, trivializes the problem of minimizing $max_i\ nt(i)$. Again, if G is not connected, the problem is infeasible. As long as G is connected, for any one-to-one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

we can achieve

$$max_i\ nt(i) = \lceil \frac{|E_Q|}{M} \rceil$$

by splitting cross terms of J equally.

< RL as the amount of communication >

A. Fixed organization

Under 'Direct Communication' assumption, whenever a cross term $x_i x_j$ was assigned, RL was increased either by one or two. Also, $x_i x_j$ can be assigned only to processors that has direct access to both DM_i and DM_j. Under 'Indirect Communication' assumption, $x_i x_j$ can be assigned to any processor that has a path in G to both DM_i and DM_j. Whenever $x_i x_j$ is assigned to DM_k, RL is increased by $nd(DM_i, DM_k) + nd(DM_j, DM_k)$, where $nd(DM_l, DM_m)$ is the distance of the shortest path between DM_i and DM_j assuming every edge has a distance 1. (nt stands for 'nominal distance'.) Therefore, when we choose RL as an objective, the problem is formulated as the following:

Minimal Message Ambulation

Given a graph G, G_Q and nt_i^*, find a decomposition that minimizes

$$\sum_k \sum_{x_i x_j \ in \ J^k} nd(DM_i, DM_k) + nd(DM_j, DM_k)$$

such that

$$nt(i) \leq nt_i^* \qquad i = 1, 2, .., M$$

(We can cast off the constraint on the capacity of each processor by letting $nt(i) = \infty, \forall i$.)

We can also consider $max_i \ nt(i)$ as our objective.

Load Balancing with Limited Message Ambulation

Given a graph G, G_Q, RL^*, find a decomposition that minimizes $max_i \ nt(i)$ such that

$$\sum_k \sum_{x_i x_j \ in \ J^k} nd(DM_i, DM_k) + nd(DM_j, DM_k) \leq RL^*$$

B. Flexible organization

When we choose RL as an objective the problem is:

Minimal Superposed Link

Given $J = x^T Q x$, nt_i^*, find a decomposition that minimizes RL such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

When we choose $max_i\ nt(i)$ as an objective, the problem is:

Load Balancing with Limited Superposed Link

Given $J = x^T Q x$, RL^*, find a decomposition that minimizes $max_i\ nt(i)$ such that $RL \leq RL^*$.

In both problems, the solution turns out to be identical to the case of 'Direct Communication'. Even though we allow indirect communication, the best route of the message flow for each cross term is a direct link in order to make RL small.

C. Semi–flexible organization

As in the case of 'Fixed organization', RL is

$$\sum_k \sum_{x_i x_j \ in \ J^k} nd(\sigma(x_i), \sigma(x_k)) + nd(\sigma(x_j), \sigma(x_k)),$$

once the one-to-one mapping

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

is determined.

Mapping for Minimal Message Ambulation

Given G_Q, $G = (V, E)$, and nt_i^* for $i = 1, 2, .., M$, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize

$$\sum_k \sum_{x_i x_j \ in \ J^k} nd(\sigma(x_i), \sigma(x_k)) + nd(\sigma(x_j), \sigma(x_k))$$

such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

Mapping for Load Balancing with Limited Message Ambulation

Given G_Q, $G = (V, E)$, and RL^*, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize $max_i \ nt(i)$ under the constraint,

$$\sum_k \sum_{x_i x_j \ in \ J^k} nd(\sigma(x_i), \sigma(x_k)) + nd(\sigma(x_j), \sigma(x_k)) \leq RL^*$$

2.3 Summary

The problem is to find a decomposition $J = \sum_i J^i$, given a quadratic function $J = x^T Q x$ so that the objectives:

1. Balance of decomposition
2. Reducing the amount of communication

are satisfied.

We can view this problem as a combinatorial optimization problem. A decomposition $J = \sum_i J^i$ can be viewed as a mapping from the set of cross terms in J to the set of processors. * (Figure 2-1) The total number of such mappings are finite (less than or equal to $M^{|E_Q|}$ where M is the number of processors, and $|E_Q|$ is the number of cross terms).

The summary of these twelve cases are in the following charts. In the following chapters, we will discuss nontrivial problems formulated in this chapter.

---

* One may imagine splitting a cross term and assign pieces to different processors. For example,

$$J = x_1^2 + x_2^2 + x_3^2 + 3x_1x_2 + 3x_2x_3 + 3x_3x_1$$

$$J^1 = x_1^2 + x_1x_2$$

$$J^2 = x_2^2 + 2x_1x_2 + 3x_2x_3$$

$$J^3 = x_1^2 + 3x_3x_1$$

However, by this type of splitting, we cannot improve any of $nt(i)$, TL, RL. Therefore, this type of decomposition is not considered, and a decomposition can be viewed as a mapping explained above.

$$J = 10x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1x_2 + 2x_2x_3 + x_3x_4 + x_4x_1$$

$$J^1 = 10x_1^2 + x_1x_2 + 2x_2x_3$$

$$J^2 = x_2^2 + x_3x_4$$

$$J^3 = x_3^2 + x_4x_1$$

$$J^4 = x_4^2$$

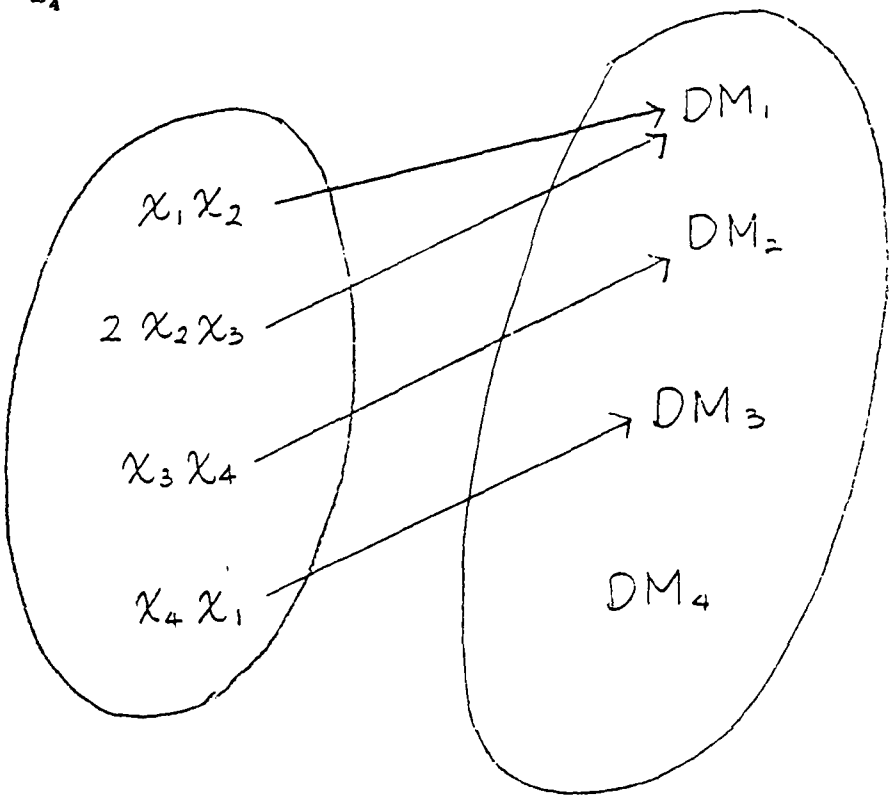


Figure 2-1 Decomposition viewed as a mapping

Summary chart for 'Direct Communication'

	TL	*RL*
Fixed Organization	Load Balancing in a Fixed Structure	Minimal Superposed Link in a Fixed Structure Load Balancing with Limited Superposed Links
Flexible Organization	Minimal Link Load Balancing with Limited Link	Minimal Superposed Link Load Balancing with Limited Superposed Link
Semi-flexible Organization	Mapping for Load Balancing	Mapping for Minimal Superposed Link Mapping for Load Balancing with Limited Superposed Link

Summary chart for 'Indirect Communication'

	TL	*RL*
Fixed Organization	Trivial	Minimal Message Ambulation Load Balancing with Limited Message Ambulation
Flexible Organization	Trivial	Minimal Superposed Link Load Balancing with Limited Superposed Link
Semi-flexible Organization	Trivial	Mapping for Minimal Message Ambulation Mapping for Load Balancing with Limited Message Ambulation

Chapter 3. DIRECT COMMUNICATION

In this chapter, the solution of nontrivial problems formulated in Chapter 2 is discussed.

3.1 TL as an amount of communication

3.1.1 Fixed organization

When an organizational structure is fixed, and a global task is given, a natural question is whether this organization is able to handle the task. This question must be answered before the issue of balancing loads is discussed. An organization's ability is determined by two criteria. First, each subdivision of an organization must be able to handle the subtask allocated to it. Second, the organizational structure must support necessary communication among subdivisions caused by coupling among subtasks. From this point of view, the question is whether there exists a proper decomposition of the given global task for this organization. A proper decomposition requires that the size of each subtask be within the capacity of the subdivision to which it is assigned. A proper decomposition also requires that if a decision of one subdivision affects the task of another subdivision, two subdivisions should have a communication link. The following formulation describes this question of organization's ability to handle the task:

Let $G_Q = (V_Q, E_Q)$ be the graph describing the structure of J, and let $G = (V, E)$ be the graph describing the fixed organizational structure. Given $G_Q = (V_Q, E_Q)$, $G = (V, E)$ (both graphs are assumed to be connected graphs) and nt_i^*, $i = 1, 2, 3, ..., M$, does there exist a decomposition

such that $nt(i) \leq nt_i^*$, $i = 1, 2, 3, ..., M$, and

such that 'Direct Communication' assumption is satisfied.

The algorithm for this problem consists of two phases. Phase 1 reduces this problem to well known Max-flow network problem. Phase 1 constructs a corresponding digraph for Max-flow problem. Phase 2 can be any algorithm that efficiently solves Max–flow problem. The following lemma will be often used to prove the algorithm.

Lemma 3-1

If a cross term $x_i x_j$ is to be assigned to DM_i or DM_j, G must have an edge (i, j). If a cross term $x_i x_j$ is to be assigned to DM_k, $k \neq i, j$, G must have edges (i, k) and (j, k).

Proof

The proof follows immediately from 'Direct Communication' assumption.

As mentioned in the end of Chapter 2, a decomposition $J = \sum_i J^i$ can be viewed as a mapping from the set of cross terms in J to the set of processors. Therefore, with the help of this lemma, the problem above can be equivalently stated as the following:

Given $G_Q = (V_Q, E_Q)$ and $G = (V, E)$, does there exist a mapping

$$\varsigma : E_Q \longrightarrow V$$

such that

if $\varsigma(i,j) = DM_k$, $k \neq i, k \neq j$, then, $(DM_i, DM_k), (DM_j, DM_k) \in E$

if $\varsigma(i,j) = DM_i$ or $\varsigma(i,j) = DM_j$, then $(DM_i, DM_j) \in E$

and such that $|\varsigma^{-1}(DM_i)| \leq nt_i^*$, $\forall i$.

The following algorithm solves this problem.

Let $M = |V|$, *the number of processors.*

Algorithm 3.1

Phase 1

1. Create $|E_Q|$ nodes corresponding to cross terms of J. Let m_{ij} denote the node corresponding to $x_i x_j$.
2. Create M nodes corresponding to processors. Let n_i denote a node corresponding to DM_i.
3. For each cross term $x_i x_j$, do

   If $(DM_i, DM_j) \in E$, make an edge from m_{ij} to n_i and make an edge from m_{ij} to n_j.

   For each neighbor of DM_i in G, say $DM_k \neq DM_j$,

   If $(DM_k, DM_j) \in E$, make an edge from m_{ij} to n_k.

   If no edge is made for $x_i x_j$, terminate with an output;

   "Organization cannot handle this task."

Let all the edges created in Step 3 have infinite capacity.

4. Create the source node s and make an edge from s to each m_{ij}. Each edge, (s, m_{ij}) created at Step 4 has capacity 1.
5. Create the sink node t and make an edge from each n_i to t. Each edge (n_i, t) created at Step 5 has capacity nt_i^*.

The digraph constructed by Phase 1 is in Figure 3-1. Let the set of edges produced in Step 3 be E_3. We can see, then, $(m_{ij}, n_k) \in E_3$ if and only if $x_i x_j$ can be assigned to DM_k without violating Lemma 3-1. If no edge is constructed for some cross term, this cross term cannot be assigned to any processor, so the organization cannot handle this task. (Line of Step 3)

Phase 2

1. Run Ford–Fulkerson labeling algorithm for the constructed digraph.
   (Set an initial feasible flow through each link that is integer. Each flow can be set zero initially.)
2. If maximum flow is $|E_Q|$, the organization can handle this task;
3. If maximum flow is less than $|E_Q|$, the organization cannot handle this task.

Let us prove this algorithm. Ford–Fulkerson algorithm [6] updates flows through an 'augmented path'. An augmentation path is a special path from the source to the sink in the undirected graph resulting from the original graph by ignoring arc directions. The augmented path satisfies the following two conditions:

1. If the direction of the original ark is the same as the direction of the augmentation path, the flow through the original arc is strictly less than the capacity of the arc.

2. If the direction of the original ark is opposite to the direction of the augmentation path, the flow through the original arc is strictly greater than zero.

Let P_a be an augmentation path at certain iteration of Ford–Fulkerson algorithm. Let e be an edge of this augmentation path P_a. Let the push flow through e be

$$push(e) = \begin{cases} capacity\ of\ ark\ -\ actual\ flow & \text{if } e \text{ has the same direction as the original ark} \\ actual\ flow & \text{if } e \text{ has the opposite direction to the original ark} \end{cases}$$

Ford–Fulkerson algorithm finds an augmentation path at each iteration and pushes flow in the direction of the augmentation path by the amount

$$\delta = min_{e \in P_a}\ push(e)$$

Since all the arc capacities of the graph in Fig 3-1 are integers, if our initial flows are integers, δ is an integer. Inductively, all the flows are kept to be integers during the run of the algorithm.

If maximum flow is $|E_Q|$, flows through all the arcs incident upon the source s must be 1 because we have $|E_Q|$ such arcs with capacity 1. Each node m_{ij} has one edge (m_{ij}, n_k) with flow 1 because of the flow conservation, and because we have shown that the flow through any arc is kept integer. This means there is a feasible assignment of the cross term $x_i x_j$ is to DM_k.

If maximum flow is less than $|E_Q|$, we claim that no assignment of cross terms can satisfy Lemma 3-1 and the capacity constraint of processors. We prove this by contradiction. Suppose there is a satisfactory assignment. For each cross term $x_i x_j$, we can find the processor DM_k to which this cross term is assigned. We can, then, set the flow through (m_{ij}, n_k) to be 1. Since the assignment of cross terms satisfies

the capacity constraint of processors, the flow through arcs into the sink t is within the capacity of these arcs. Therefore, the maximum flow is $|E_Q|$. Contradiction.

Thus, Algorithm 3.1 is proved.

Phase 1 of algorithm 1.1 takes $O(|E_Q|M)$ because for each cross term, every node should be checked if it can handle the cross term. Ford–Fulkerson algorithm takes $O(|A|)$ where A is a set of edges of the graph in Fig 3-1. $|A|$ is no greater than $|E_Q| + M + |E_Q|M$. Therefore, for our problem, Step 1 of Phase 2 takes time $O(|E_Q|M)$. Since the maximum flow is no greater than $|E_Q|$, Phase 2 takes $O(|E_Q|^2M)$. Algorithm 1.1 has running time $O(|E_Q|^2M)$. (At step 1 of phase 2, any max–flow algorithm can be used as long as they generate integral solution. If we use such an algorithm other than Ford–Fulkerson, the time complexity may be different.)

We can extend Algorithm 1.1 in order to obtain a balanced allocation of the global cost function, given a fixed processor network. Let us say that the best–balanced allocation is the one that minimizes the load of the most heavily loaded processor.

Load Balancing in a Fixed Structure

Given $G_Q = (V_Q, E_Q)$ and a graph $G = (V, E)$, find a decomposition that minimizes $max_i\ nt(i)$.

We can solve this problem by running the phase 2 of algorithm 1.1 recursively.

Algorithm 3.2

1. Run Line 1 through Line 4 of phase 1 of Algorithm 3.1

   Set $nt^* = \lceil \frac{|E_Q|}{M} \rceil$

2. Create the sink node t and make an edge from each n_i to t.

3. Set the capacity of each edge (n_i, t) to be nt^*.

4. Run Ford–Fulkerson labeling algorithm for the constructed digraph.

   (Set an initial feasible flow through each link that is integer. Each flow can be set zero initially.)

   If maximum flow is $|E_Q|$, stop;

   ($min\ max_i\ nt(i) = nt^*$)

   If maximum flow is less than $|E_Q|$, $nt^* := nt^* + 1$

   go to Line 3.

We know that the algorithm terminates before nt^* becomes greater than $|E_Q|$ because $min\ max_i\ nt(i)$ is no greater than $|E_Q|$. Therefore, the time complexity of Algorithm 3.2 is $O(|E_Q|^3 M)$. In Step 4 of Algorithm 3.2, if we do binary search of nt^* rather than increase it one at each iteration, we can run Ford–Fulkerson algorithm only $O(log\ |E_Q|)$ times. Therefore the complexity of Algorithm 3.2 can be improved to $O(M|E_Q|log|E_Q|)$.

When the fixed organizational structure happens to be a tree, we present a special algorithm in order to minimizes $max_i nt(i)$.

Fixed Tree:

Given G_Q (a graph describing a global cost function) and $T = (V, E_T)$ (a tree structured network of decision makers), find a decomposition that minimizes $max_i\ nt(i)$.

The following lemma is used in order to analyze the algorithm.

Lemma 3-2

Suppose the organizational structure is a tree, and DM_i is responsible for x_i.

1. If DM_i and DM_j are directly linked, i.e. $(DM_i, DM_j) \in E_T$, the cross term $x_i x_j$ is assigned either to DM_i or to DM_j.
2. If DM_i and DM_j are connected by exactly two hops, i.e. there exists DM_k such that $(DM_i, DM_k), (DM_j, DM_k) \in E_T$ and $(DM_i, DM_j) \notin E$, $x_i x_j$ is assigned to DM_k.

Proof

1. Suppose DM_i and DM_j are directly linked, and $x_i x_j$ is assigned to DM_k, $k \neq i, j$. J^k depends upon x_i and x_j. Therefore, by 'Direct Communication' assumption , $(DM_i, DM_k) \in E_T$ and $(DM_j, DM_k) \in E_T$. Therefore, DM_i, DM_j, and DM_k form a cycle. Contradiction.
2. Suppose DM_i and DM_j are connected through DM_k. If $x_i x_j$ is assigned to DM_i or DM_j, by 'Direct Communication' assumption, $(DM_i, DM_j) \in E_T$. Therefore DM_i, DM_j, DM_k form a cycle. Contradiction. If $x_i x_j$ is assigned to DM_l, $l \neq i, j, k$. By 'Direct Communication' assumption , $(DM_i, DM_l) \in E_T$ and $(DM_j, DM_l) \in E_T$. Therefore, DM_i, DM_j, DM_k and DM_l form a cycle. Contradiction.

   Q.E.D.

Algorithm 3.3

Given a tree, $T = (V, E_T)$ and $J(x_1, x_2, ..., x_M)$,

Step 1. Check if the given tree can handle the cost function. In other words, for all cross terms $x_i x_j$, check if DM_i and DM_j are within two hops.

Step 2. Assign each square term x_i^2 in J^i.

For each cross term $x_i x_j$, if there exist DM_k such that $(DM_i, DM_k) \in E_T$, $(DM_j, DM_k) \in E_T$, $x_i x_j$ is assigned in J^k

Step 3. Let us define

$nec(i) \equiv$ the number of terms necessarily assigned to DM_i by step 2.

$mx \equiv max_i\ nec(i)$

$L(G)$, *leaves* $\equiv$ a set of nodes of a graph G whose degree is 1.

$B(G)$ $\equiv$ a set of edges incident upon the nodes in $L(G)$

$T(n) = (V(n), E_T(n))$

$$\equiv \begin{cases} T & \text{if } n = 1 \\ (V(n-1) - L(T(n-1)) \ , \ E_T(n-1) - B(T(n-1))) & \text{if } n \geq 2 \end{cases}$$

For iteration n from 1 to r where $T(r) = \emptyset$, do the following:

for all pairs i, j such that $DM_i \in L(T(n))$, $DM_j \in T(n)$, and $(DM_i, DM_j) \in E(n)$

a) if $nec(i) < mx$,

$x_i x_j$ is assigned to DM_i

$nt(i) = nec(i) + 1$; frozen

b) if $nec(i) = mx$, and $nec(j) < mx$,

$x_i x_j$ is assigned to DM_j

$nt(i) = nec(i)$; frozen

$nec(j) := nec(j) + 1$

c) if $nec(i) = mx$, and $nec(j) = mx$,

$x_i x_j$ is assigned to DM_i

$nt(i) = nec(i) + 1 = mx + 1$; frozen

(The terminology 'frozen' means that no more cross term can be assigned to the processor; therefore, $nt()$ is set.)

Explanation:

Step 1. For some trees, no decomposition can satisfy our assumptions. See Figure 3-2 as an example.

Step 2. This step assigns to each DM_i all the terms that J^i must necessarily have according to to the second point of Lemma 3-2. (At the end of Step 2, we are left with at most $M-1$ cross terms to assign. These cross terms are characterized as $x_i x_j$ such that $(DM_i, DM_j) \in E_T$.)

Step 3. At each iteration, we take leaves and their neighbors (A *leave* is defined as a node whose degree is 1) and eliminates leaves with their incident edges. The number of cross terms assigned to these leaves are frozen when they are removed. Step 3 is a strategy to assign cross terms corresponding to the edges of the tree.

Theorem 3.3

Algorithm 3.3 minimizes $max_i\ nt(i)$, for a fixed tree in polynomial time, and $mx \leq max_i\ nt(i) \leq mx + 1$

Proof

First, we claim that at any iteration n, $nec(i) \leq mx$ for any leaf DM_i as long as $nt(i)$ has not been frozen. Suppose not. Let DM_i be the first node in the progression of Algorithm 3.3 such that DM_i is a leaf and $nec(i) > mx$. Since $nec(k) \leq mx$ for all processors DM_k before Step 3, the only way it can happen is $nec(i) = mx$ at a certain time n, and a cross term $x_i x_l$ is assigned to DM_i. DM_l must be a leaf incident on DM_i when this happens, and this DM_l is frozen as the assignment of $x_i x_l$ is made. (Figure 3-3a) At this time n, $nec(l) \leq mx$ because we defined DM_i to be the first node such that *nec* exceeds mx while bing a leaf. Since $nec(i)$ is mx, and DM_l is a leaf incident upon DM_i, $x_i x_l$ cannot be assigned to DM_i. Contradiction.

Secondly, if no incidence of Step 3-c happens, $nec(k) \leq mx$ for each processor

DM_k. Therefore, $max_k\ nt(k) = mx$.

Finally, if Step 3-c happens at some iteration n (Figure 3-3b), we know from the first claim that $max\ nt(k) = mx + 1$ at the end of the algorithm. We claim that $min\ max_k\ nt(k)$ is indeed $mx + 1$ for this global function. Let us say that DM_i is a leaf with $nec(i) = mx$, and DM_j is its neighbor with $nec(j) = mx$. If $nec(i)$ and $nec(j)$ are both originally mx before Step 3 begins, obviously $max_k\ nt(k) \geq mx + 1$, because $x_i x_j$ must be assigned to either DM_i or DM_j by Lemma 3-2. Therefore, $max_k\ nt(k) = mx + 1$. Even though $nec(i) < mx$ or $nec(j) < mx$ before Step 3, we cannot make $nec(i)$ or $nec(j)$ less than mx at time n with keeping $nt(k) \leq mx$ for all k. For example, if $nec(i) < mx$ before Step 3, some cross terms of the form $x_i x_l$ must have been assigned to DM_i at certain iterations before n. Take an arbitrary, such l, and say $x_i x_l$ has been assigned to DM_i at iteration $n_l < n$. Therefore, at iteration n_l, DM_l is a leaf incident on DM_i, and $nec(l) = mx$ (Figure 3-3c). If $nec(l)$ is mx before Step 3, $x_i x_l$ must be assigned to DM_i as long as we try to make $max_k\ nt(k)$ less than $mx + 1$. If this is the case for all l, $nec(i)$ has to be mx at time n in order to keep $nt(k) \leq mx$ for all k. If for some l, say l_1, $nec(l_1) < mx$ before Step 3, there must have been some cross terms of the form $x_{l_1} x_p$ that has been assigned to DM_{l_1} at some iteration $n_p < n_l$. Therefore, at iteration n_p, DM_p is a leaf incident on DM_{l_1}, and $nec(p) = mx$ (Figure 3-3d) If $nec(p)$ is mx before Step 3, $x_{l_1} x_p$ must be assigned to DM_{l_1} as long as we try to make $max_k\ nt(k)$ less than $mx + 1$. If this is the case for all p, $nec(l)$ has to be mx at time n_p in order to keep $nt(k) \leq mx$ for all k. If for some p, say p_1, $nec(p_1) < mx$ before Step 3, there must have been some cross terms of the form $x_{p_1} x_q$ that has been assigned to DM_{p_1} at some iteration $n_q < n_p$, and $nec(q) = mx$ at iteration n_p. If we keep repeating this argument, we will eventually get to a node DM_* such that $nec(*)$ is mx before Step 3, because the graph G is a finite tree. Therefore, $nec(i)$ must be mx at iteration n

in order to keep $nt(k) \leq mx$ for all k. Therefore $max_k\ nt(k) \geq mx + 1$. Since we get $max_k\ nt(k) = mx + 1$ at the end of Algorithm 3.3, $max_k\ nt(k) = mx + 1$.
Q.E.D.

3.1.2 Flexible organization

Minimal Link

Given $G_Q = (V_Q, E_Q)$, nt_i^*, find a decomposition that minimizes TL such that

$$nt(i) \leq nt_i^*, \qquad i = 1, 2, ..., |V_Q|$$

Load Balancing with Limited Link

Given $G_Q = (V_Q, E_Q)$, tl^*, find a decomposition that minimizes $max_i\ nt(i)$ such that $TL \leq TL^*$.

The recognition version (discussion problem) of these two problems are identical.

INSTANCE

$G_Q = (V_Q, E_Q)$, $M = |V_Q|$

tl, total number of necessary links

nt^*, maximum of $nt(i)$ over $i = 1, 2, ..., M$

PROBLEM

Does there exist a decomposition with total number of necessary links, tl and $max_i\ nt(i) = nt^*$?

No efficient algorithm for 'Minimal link' or 'Load Balancing with Limited link' has been found. We conjecture that the recognition version of these problems is NP–complete.

An efficient algorithm can be designed, however, for a special case of 'Minimal Link' problem. When the global cost function is $J = x^T Q x$, where Q is bandwidth limited matrix, a dynamic programming can be used, provided that some additional constraint is imposed on the assignment of existing cross terms.

Minimal Link for bandwidth limited cost

Given $J = x^T Q x$, where Q is bandwidth limited by a fixed integer W, ($Q(i,j) = 0$ if $|i-j| \geq W$)

find a decomposition $J = \sum_k J^k$ that minimizes TL subject to the following constraints:

Constraint 1 : $nt(i) \leq nt_i^*$

Constraint 2 : any cross term $x_i x_j$ $(i < j)$ can be only in subcost function J^k,

where k satisfies $j - W + 1 \leq k \leq j$. (See Figure 3-4)

The special feature of this problem is that the interaction between variables is local. If two variables x_i, x_j are sufficiently distant ($|i-j| \geq W$), two processors (or divisions of an organization) responsible for these variables can make decisions independent of each other. Here is the reason. Without loss of generality, let us assume $i < j$. For any cross term of the form $x_i x_\omega$ in the cost function J, $|i-\omega| < W$. If $i < \omega < j$, $x_i x_\omega$ can be assigned to processors $DM_{\omega-W+1}$, $DM_{\omega-W+2}$,...,DM_ω, but not DM_j. (Figure 3-5a) If $\omega < i < j$, $x_\omega x_i$ can be assigned to processors

DM_{i-W+1}, DM_{i-W+2},...,DM_i, but not DM_j. (Figure 3-5b) Therefore, no cross term of the form $x_i x_\omega$ can be in J^j. By the same token, no cross term of the form $x_j x_\omega$ can be in J^i. Therefore, J^i does not depend upon x_j, and J^j does not depend upon x_i. Dynamic programming takes advantage of this feature of the locality of interaction.

Decomposing a cost function can be viewed as assigning cross terms to processors. Let us break up this labor of assigning cross terms into M stages, where M is the number of variables. At each stage $t \in \{1, 2, 3, ..., M\}$, we assign all the cross terms of the form

$$x_l x_t, \quad t - W + 1 \leq l < t.$$

Because of Constraint 2, these cross terms can only be assigned to

$$DM_{t-W+1},\ DM_{t-W+2},\ ....,\ , DM_t$$

(Figure 3-4) Therefore, the number of choices for a decision at each stage is at most $W(W-1)$ (W processors to choose for each of at most $W-1$ cross terms of the form $x_l x_t$, $t - W + 1 \leq l < t$). This gives hopes for polynomial–time dynamic programming because W is fixed. A state at each stage should contain the number of cross terms assigned to each processor up to that stage. A state also includes the location of communication links required by the assignments of cross terms up to that stage.

Let us describe the dynamic programming rigorously. Let the vector $\tilde{U}(t) \in Z^W$ describe the number of terms assigned to processors DM_{t-W+1}, DM_{t-W+2}, ..., DM_t

until the beginning of step t. More precisely,

$$\tilde{U}(t) = \begin{pmatrix} U_1(t) \\ \vdots \\ U_i(t) \\ \vdots \\ U_W(t) \end{pmatrix}$$

$$U_i(t) = \begin{cases} \textit{the number of cross terms} \\ \textit{assigned to } DM_{t-W+i} \\ \textit{until the beginning of step } t & \text{if } t-W+i \geq 1 \\ 0 & \text{if } t-W+i \leq 0 \end{cases}$$

Notice that only the cross term of the form $x_l x_s$, $l < s < t$ is assigned to processors until the beginning of step t. This kind of cross term cannot be assigned to processor DM_t from constraint 2. Therefore, DM_t does not have any cross term at the beginning of step t. Consequently, $U_W(t) = 0$ for all $t = 1, 2, ..., M$. Also, since no assignment has been made at the beginning of step 1, $\tilde{U}(1) = \tilde{0}$. (Figure 3-6)

Let the symmetric matrix $\tilde{V}(t) \in \{0,1\}^{W \times W}$ describe the link structure necessary for the assignment done until the beginning of step t. More precisely, $\tilde{V}(t)$ is a upper diagonal matrix such that

for $1 \leq i < j \leq M$

$$V_{ij}(t) = \begin{cases} 1 & \text{if link } (DM_{t-W+i}, DM_{t-W+j}) \\ & \text{is necessary for cross terms} \\ & \text{assigned until the beginning of step } t \\ 0 & \text{otherwise} \end{cases}$$

Notice that $V_{iW} = 0$ for $i = 1, 2.., W-1$, because cross term involving x_t has not been assigned until the beginning of step t. Also, $\tilde{V}(0) = 0$ because no assignment has been made at the beginning of step 1, so no link is necessary.

Let $\tilde{d}(t) \in \{0,1,2,...,W\}^{(W-1)}$ describe a decision made at stage t.

$$\tilde{d}(t) = \begin{pmatrix} d_1(t) \\ \vdots \\ d_i(t) \\ \vdots \\ d_{W-1}(t) \end{pmatrix}$$

This decision $d_i(t)$ is where to assign a cross term of the form

$$x_{t-W+i}x_t, \quad i = 1,2,...,W-1$$

$$d_i(t) = \begin{cases} 0 & \text{if } x_{t-W+i}x_t \text{ is not in } J \\ k & \text{if } x_{t-W+i}x_t \text{ is assigned to } DM_{t-W+k} \end{cases}$$

$$i = 1,2,3,...,W-1 \qquad k = 1,2,3,...,W$$

Now we are ready to write an equation that describes the state evolution. Let us define a vector $\tilde{C}(\tilde{d}(t)) \in Z^W$:

$$C_i(\tilde{d}(t)) = \text{ the number of components having the number i in } \tilde{d}(t)$$

$C_i(\tilde{d}(t))$ is the number of cross terms newly added to DM_{t-W+i}.

The evolution of $\tilde{U}(t)$ is

$$U_i(t+1) = \begin{cases} U_{i+1}(t) + C_i(\tilde{d}(t)) & \text{for i=1,2,...,W-1} \\ C_W(\tilde{d}(t)) & \text{for i=W} \end{cases}$$

(See Figure 3-7)

$V_{ij}(t+1)$ tells whether there is a link between $DM_{t+1-W+i}$ and $DM_{t+1-W+j}$ at the beginning of step $t+1$. Let us consider the case $i < j < W-1$, first. (See Figure 3-8a) If there is to exist link $(DM_{t+1-W+i}, DM_{t+1-W+j})$, $J^{(t+1-W+i)}$ should have

a term involving $x_{t+1-W+j}$, or $J^{(t+1-W+j)}$ should have a term involving $x_{t+1-W+i}$. Since cross terms of the form

$$x_l x_t, \quad t - W + 1 \leq l < t$$

are assigned at step t, the only way that this link is newly added at step t is the following:

1) $x_{t+1-W+i}x_t$ is assigned to $DM_{t+1-W+j}$ or

2) $x_{t+1-W+j}x_t$ is assigned to $DM_{t+1-W+i}$.

Now, let us consider the case $i < j = W - 1$. (See Figure 3-8b) In this case $t + 1 - W + j = t$. If there is to exist link $(DM_{t+1-W+i}, DM_t)$, $J^{(t+1-W+i)}$ should have a term involving x_t, or J^t should have a term involving $x_{t+1-W+i}$. Since cross terms of the form

$$x_l x_t, \quad t - W + 1 \leq l < t$$

are assigned at step t, the only way that this link is newly added at step t is the following:

1) $x_{t+1-W+i}x_t$ is assigned to $DM_{t+1-W+j} = DM_t$ or

2) $x_l x_t$ is assigned to $DM_{t+1-W+i}$ for some $t - W + 1 \leq l < t$.

For $i < j = W$, as mentioned before, $V_{ij}(t+1) = 0$.

for $i = 1, 2, .., j - 1$

$$V_{ij}(t+1)$$
$$= \begin{cases} V_{(i+1)(j+1)}(t) \vee (d_{i+1}(t) = j+1) \vee (d_{j+1}(t) = i+1) & \text{for } j = 2 \text{ to } W-2 \\ (d_{i+1}(t) = W) \vee (d_k(t) = i+1, \ \textit{some } 1 \leq k \leq W-1) & \text{for } j = W-1 \\ 0 & \text{for } j = W \end{cases}$$

where $\vee$ is Boolean 'OR'.

Let us define

$$cost(\tilde{V}(t),\ \tilde{d}(t)\ ) = \textit{the number of newly added links at step } t$$

then, $tl = \sum_{t=1}^{M} cost(\tilde{V}(t),\ \tilde{d}(t)\ )$.

$(\tilde{U}(t),\ \tilde{V}(t)\ )$ is a state, and the state space is

$$X = \{(\tilde{U},\ \tilde{V}) | U_i \leq nt_i,\ i = 1, 2, ..., M\}$$

Now we can describe 'Minimal Link for bandwidth limited cost' in the framework of dynamic programming.

$$U_i(t+1) = \begin{cases} U_{i+1}(t) + C_i(\tilde{d}(t)) & \text{for i=1,2,...,W-1} \\ C_W(\tilde{d}(t)) & \text{for i=W} \end{cases}$$

$$V_{ij}(t+1)$$

$$= \begin{cases} V_{(i+1)(j+1)}(t) \vee (d_{i+1}(t) = j+1) \vee (d_{j+1}(t) = i+1) & \text{for } j = 2 \text{ to } W-2 \\ (d_{i+1}(t) = W) \vee (d_k(t) = i+1,\ \textit{some}\ 1 \leq k \leq W-1) & \text{for } j = W-1 \\ 0 & \text{for } j = W \end{cases}$$

$$(U,V)(1) = 0, \quad (U,V)(t) \in X$$

$$\textit{minimize} \sum_{t=1}^{M} cost(\tilde{V}(t),\ \tilde{d}(t)\ )$$

We can solve this problem using the dynamic programming algorithm.[10]

If we restrict TL^* to be $M-1$ (In other words the organizational structure is a tree.), some analytical statements can be made concerning 'Load Balancing with

Limited Link'. The following problem represents a situation where an organization with M agents cannot afford to build more than $M-1$ communication links.

Load Balancing for tree structure

Given $J = x^T Q x$, find a decomposition that minimizes $max_i nt(i)$ such that $TL \leq M-1$.

or equivalently,

Given $J = x^T Q x$, find a decomposition that minimizes $max_i nt(i)$ such that $TL = M-1$.

We assume that G_Q is connected. Therefore, it immediately follows from Lemma 2-1 that the resulting graph G must be connected for any decomposition. Therefore, the constraint $TL \leq M-1$ is equivalent to the constraint $TL = M-1$.

Some analytical properties of these problems have been studied.

Definition 3-4

A centralized tree , $T_c = (V, E), |V| = M$ is a tree that has one node DM_i with $d(DM_i) = M-1$ and $M-1$ nodes with degree 1. We call the node with degree, $M-1$ **center.**

Lemma 3-5

For a graph $G = (V, E)$, if all the pairs of nodes in V are within two hops from one another, and G has no cycle, it forms a centralized tree.

Theorem 3-6

In the 'Load Balancing for tree structure' problem, if G_Q has a clique of size c,

$max_i\ nt(i) \geq \binom{c-1}{2}$.

Proof

1) When the whole graph, G_Q is a clique of size $c = M$:

   any pair of nodes of G have to be within two hops from each other. (Lemma 2-1) Therefore, the organizational structure is a centralized tree. (Lemma 3-5) Let DM_i be the center, then all the $\binom{c-1}{2}$ cross terms between $c-1$ neighbors must be assigned in J^i. (Lemma 3-2) Therefore, $nt(i) \geq \binom{c-1}{2}$ Q.E.D.

2) When G_Q has a clique of size $c < M$:

   let S be a subgraph of G that consists of c nodes corresponding to the clique. If all the nodes in S are within two hops from one another, the argument is reduces to 1). If there exists a pair, DM_i, DM_j in S such that DM_i and DM_j are connected through DM_k not in S, all c nodes in S must have a direct link to DM_k; otherwise, a node not having a direct link to DM_k has more than two hops to either DM_i or DM_j. Therefore, DM_k is a center of a centralized tree of size $c+1$. J^k must have at least $\binom{c}{2}$ Q.E.D.

An interesting observation about this problem is the following:
Given G_Q and the constraint, $TL = M - 1$, it is possible for a tree that is not a subgraph of G_Q to minimize $max_i\ nt(i)$ over all possible trees.

This is illustrated in Figure 3-9. This fact indicates that an organization can minimize the load of the highest–loaded division under a strong constraint on the number of communication links by introducing a link between divisions not directly influencing one another.

Let us now observe some theoretical results for special cost functions. The

following lemma will be used many times for analytical discussion of 'Load Balancing for tree structure'.

Lemma 3-7

Suppose the global cost function J has cross terms x_ix_j, x_jx_k, and x_kx_i. If there exists a link between DM_i and DM_l, and between DM_j and DM_l in the graph G, $l \neq i,j,k$, (Figure 3-10a) cross terms x_ix_k and x_jx_k must be assigned to DM_l as well as cross term x_ix_j. Therefore, a link between DM_l and DM_k is necessary.

Proof

From Lemma 3-2, x_ix_j must be assigned to DM_l.

Suppose x_ix_k is assigned to DM_i or DM_k, then there exists a link between DM_i DM_k. Therefore, DM_j and DM_k are three hops apart because there is a unique path between any pair of nodes in a tree. (Figure 3-10b) This contradicts Lemma 2-1 because $Q(j,k) \neq 0$.

Suppose x_ix_k is assigned to DM_j. Then, there exists a link between DM_i and DM_j by 'Direct Communication' assumption, so DM_i, DM_j, DM_l form a loop. This contradicts the tree constraint.

Suppose x_ix_k is assigned to $DM_{l'}$, $l' \neq i, l' \neq j, l' \neq l$. In this case, there exists a link between DM_i and $DM_{l'}$, DM_k and $DM_{l'}$. Therefore, DM_i and DM_k are four hops apart. (Figure 3-10c) This again contradicts Lemma 2-1. Therefore, x_ix_k must be assigned to DM_l.

Q.E.D.

Let the global cost function be described by $G_Q = (V_Q, E_Q)$. We discuss the cases of special cost functions such that G_Q has a subgraph $S = (V_S, E_S)$ of the

form

$$V_S = \{i, 1, 2, ..., n\} \tag{3-1}$$

$$E_S = \{(i,j)|j = 1,2,...,n\} \cup \{(j,j+1)|j = 1,2,...,n-1\} \tag{3-2}$$

(See Figure 3-11)

Lemma 3-8

Assume that G_Q has the structure introduced by equation (3-1), (3-2). For $j = 1,2,...,n-1$, if x_ix_j is assigned to DM_k, ($k \neq i$ and $k \neq j$, $j = 1,2,..,n$) then, $x_ix_{(j+1)}$ must also be assigned to DM_k.

For $j = 2,...,n$, if x_ix_j is assigned to DM_k, ($k \neq i$ and $k \neq j$, $j = 1,2,..,n$) then, $x_ix_{(j-1)}$ must also be assigned to DM_k.

Proof

i) For $j = 2,...,n$, Since x_ix_j is in DM_k, there exists a link between DM_i and DM_k, DM_j and DM_k in G, the resulting organizational structure (Direct Communication assumption). Therefore, by Lemma 3-7, x_ix_{j+1} must be assigned to DM_k.

ii For $j = 1,2,...,n-1$, the same argument shows that x_ix_{j-1} must be assigned to DM_k.

Theorem 3-9

Assume that G_Q has the structure introduced by equation (3-1), (3-2). If x_ix_j (j is any of $\{1,2,...,n\}$) is assigned to DM_k where $k \neq i, k \neq j, j = 1,2....,n$, cross terms x_ix_j, $j = 1,2,..,n$ are assigned to DM_k. The resulting structure of an organization must contain the following edges:

$$\{(DM_k, DM_i)\} \cup \{(DM_k, DM_j)|j = 1,2,...,n\}$$

Proof

By recursively applying the result of Lemma 3-8, we can easily see that x_ix_j must be assigned to DM_k for all x_ix_j $j = 1, 2, ..., n$ if one of them is assigned to DM_k. Therefore, from 'Direct Communication' assumption, the resulting graph must contain

$$\{(DM_k, DM_i)\} \cup \{(DM_k, DM_j) | j = 1, 2, ..., n\}$$

Theorem 3-10

Assume that G_Q has the structure introduced by equation (3-1), (3-2). Suppose there is a link between DM_i and DM_j for some $j < n$ in G. (x_ix_j must be assigned either to DM_i or DM_j.)

1) If x_ix_{j+1} is assigned to DM_j, cross terms, x_ix_{j+2}, x_ix_{j+3},..., x_ix_n must all be assigned to DM_j.

2) If x_ix_{j-1} is assigned to DM_j, cross terms, x_ix_{j-2}, x_ix_{j-3},..., x_ix_1 must all be assigned to DM_j.

Proof

1) Since x_ix_{j+1} is assigned to DM_j, there is a link between DM_j and DM_{j+1}. (See Figure 3-12) The global cost J has x_ix_{j+2}, $x_{j+1}x_{j+2}$,$x_{j+2}x_i$.
   Thus, by Lemma 3-7, x_ix_{j+2} must be assigned to DM_j
   Consequently, there exists a link between DM_{j+2} and DM_j. By using the same argument recursively, we can show that x_ix_{j+3},....,x_ix_n must all be assigned to DM_j.
2) Same proof as 1) in reverse direction

Now we observe the case where $G_Q = (V_Q, E_Q)$ is of the form

$$V_Q = \{\ i, 1, 2, 3, .., n\ \} \tag{3-3}$$

$$\begin{aligned} E_Q = &\{(DM_i, DM_j) | j = 1, 2, 3..., n\} \cup \\ &\{(DM_j, DM_{j+1}) | j = 1, 2, 3, .., n-1\} \cup \\ &\{(DM_n, DM_1)\} \end{aligned} \tag{3-4}$$

(Figure 3-13)

The results of Lemma 3-8 and Theorem 3-9 can be directly applied to this case because the graph $S = (V_S, E_S)$ described in eqn 3-1 and eqn 3-2 is a subgraph of G_Q described by eqn 3-3 and eqn 3-4. Theorem 3-10 can be modified as follows:

Theorem 3-11

Consider G_Q described by eqn (3-3) and (3-4). Suppose there is a link between DM_i and DM_j for some $1 \leq j \leq n$ in G. ($x_i x_j$ must be assigned either to DM_i or DM_j.) If $x_i x_{j'}$, where $(j, j') \in E_Q$, is assigned to DM_j, all the cross terms of the form $x_i x_l$ $l = 1, 2.., n$, must be assigned to DM_j.

Proof

For the sake of concise explanation, let us define

$$rn(j) = (j+1)\ mod\ n \qquad (meaning\ 'right\ neighbor')$$

$$ln(j) = (j-1)\ mod\ n \qquad (meaning\ 'left\ neighbor')$$

then, $j' = rn(j)$ or $ln(j)$.

i) Suppose $x_i x_k$ is assigned to DM_j for any $k \neq ln(j)$. We claim that $x_k x_{rn(k)}$ must also be assigned to DM_j. From 'Direct Communication' assumption, there exists a link between DM_j and DM_k. (See Figure 3-14) The global cost

function J has $x_i x_k$, $x_k x_{rn(k)}$, $x_{rn(k)} x_i$. Therefore, by Lemma 3-7, $x_i x_{rn(k)}$ must be assigned to DM_j.

If $x_i x_{rn(j)}$ is assigned to DM_j, all the cross term of the form

$$x_i x_l, \quad l = (j+2) \ mod \ n \ \ , l = (j+3) \ mod \ n \ \ , ..., l = (j+n-1) \ mod \ n$$

are assigned to DM_j by induction.

ii) Suppose $x_i x_k$ is assigned to DM_j for any $k \neq rn(j)$. We can show that $x_k x_{ln(k)}$ must also be assigned to DM_j by the same argument in the opposite direction. Therefore, by induction, if $x_i x_{ln(j)}$ is assigned to DM_j, all the cross term of the form

$$x_i x_l, \quad l = (j-2) \ mod \ n \ \ , l = (j-3) \ mod \ n \ \ , ..., l = (j-n+1) \ mod \ n$$

are assigned to DM_j.

Q.E.D.

From this theorem we can also conclude the following:

If there is a link between DM_i and DM_j for some $1 \leq j \leq n$ in G, and $x_i x_{j'}$, where $(j, j') \in E_Q$, is assigned to DM_j, the resulting graph G is a centralized tree with the center DM_j.

Lemma 3-12

If $x_i x_j$ is assigned to $DM_{j'}$, $j' \neq i, j$, the resulting graph G is a centralized tree with the center $DM_{j'}$.

Proof

i) $j' = ln(j)$ or $rl(j)$:

There must be a link between DM_i and $DM_{j'}$ by 'Direct Communication'

assumption, and $x_i x_j$ is assigned to $DM_{j'}$. Therefore, by Theorem 3-11, all the cross term of the form

$$x_i x_l, \quad l = 1, 2, ..., n$$

are assigned to $DM_{j'}$. Therefore, the resulting graph is a centralized tree with the center $DM_{j'}$.

ii) $j' \neq ln(j)$ and $j' \neq rl(j)$:

Since $x_i x_j$ is assigned to $DM_{j'}$, there is a link between DM_i and $DM_{j'}$, and DM_j and $DM_{j'}$.

The global cost function J has $x_i x_j$, $x_j x_{rn(j)}$,$x_{rn(j)} x_i$, so by Lemma 3-7 $x_i x_{rn(j)}$ must be assigned to $DM_{j'}$. Therefore, there must be a link between $DM_{j'}$ and $DM_{rn(j)}$. By recursively applying Lemma 3-7, we can see that all the cross term of the form

$$x_i x_l, \quad l = (j+2) \ mod \ n \ \ , l = (j+3) \ mod \ n \ \ , ..., l = ln(j')$$

are assigned to $DM_{j'}$.

The global cost function J has $x_i x_j$, $x_j x_{ln(j)}$,$x_{ln(j)} x_i$, so by Lemma 3-7 $x_i x_{ln(j)}$ must be assigned to $DM_{j'}$. Therefore, there must be a link between $DM_{j'}$ and $DM_{ln(j)}$. By recursively applying Lemma 3-7, we can see that all the cross term of the form

$$x_i x_l, \quad l = (j-2) \ mod \ n \ \ , l = (j-3) \ mod \ n \ \ , ..., l = rn(j')$$

are assigned to $DM_{j'}$.

Therefore, the resulting graph is a centralized graph with the center $DM_{j'}$.

Q.E.D.

Theorem 3-13

For a global cost function described by G_Q in eqn (3-3) and (3-4), with $n \geq 3$, the

optimally balanced decomposition under 'tree constraint' gives

$$min_{tree}\ max_i\ nt(i) = n$$

and the resulting graph is a centralized tree with the center DM_i.

Proof

i) Suppose for some $j \in \{1, 2, ..., n\}$, $x_i x_j$ is assigned to DM_k, $k \neq i, j$. By Lemma 3-12, the resulting organizational structure is a centralized tree with the center DM_k. (Figure 3-15a) For all $l \neq k, i$, DM_i and DM_l are two hops apart, and DM_k is the connector. Therefore, by Lemma 3-2, $n-1$ cross terms of the form $x_i x_l$, $l \neq k, i$, must necessarily be assigned to DM_k. Also, for

$$l = (k+1)\ mod\ n,\ (k+2)\ mod\ n, ..., (k+n-2)\ mod\ n$$

DM_l and $DM_{rn(l)}$ are two hops apart, and DM_k is the connector. Therefore, cross terms of the form $x_l x_{rn(l)}$ must be assigned to DM_k for

$$l = (k+1)\ mod\ n,\ (k+2)\ mod\ n, ..., (k+n-2)\ mod\ n$$

Consequently J^k must have at least $2n-3$ cross terms. Thus, $nt(k) \geq 2n-3$.

ii) Suppose for all $j = 1, 2, ..., n$, $x_i x_j$ is assigned to DM_i or DM_j. By Lemma 3-1, there must exist a link between DM_i and DM_j for $j = 1, 2.., n$ Therefore, the resulting organizational structure is a centralized tree with the center DM_i. (Figure 3-15b) For all $j = 1, 2, .., n$, DM_j and $DM_{rn(j)}$ are two hops apart, and DM_i is a connector. Therefore, by Lemma 3-2, cross terms of the form $x_j x_{rn(j)}$, $j = 1, 2, .., n$ must be assigned to DM_i. If we assign cross terms of the form $x_i x_j$ to DM_j for $j = 1, 2, .., n$, J^i ends up having n cross terms, and J^j ends up having one cross term, for $j = 1, 2, .., n$.

$$nt(j) = 1, \quad j = 1, 2, ..., n$$

$$nt(i) = n$$

Therefore,

$$max\ nt = n$$

From i) and ii)

$$min\ max\ nt = n$$

Q.E.D.

Concerning analytic discussions of the relationship between flexible organizational structure and its global cost function, there is a lot of room for research. The following open questions are suggested for future research.

Does there always exist a tree contained in G_Q which minimizes $max_i nt(i)$ over all trees?

As for an organizational task, what more can be said about G_Q in between two extremes of strong connectivity and diagonality?

How can we define a measure of intricacy of the task? One idea of defining a coupling measure is

$$\frac{1}{max_{(i,j)}\ shortest\ path\ between\ i\ and\ j\ in\ G_Q}$$

We want to examine other ways of defining a measure of coupling. We expect that $min\ max_i\ nt(i)$ increases as a measure of coupling in the task increases under a constrained tl. We also want to develop some criteria according to which one can check a cost function (represented by G_Q) to see if decentralized computation results in better speed of convergence.

3.1.3 Semi–flexible organization

So far, we have implicitly assumed that DM_i is in charge of the value of x_i. We are also interested in the following problem where the organization designer has freedom to mandate which division (DM's) is responsible for which decision (x_i's), but communication structure of an organization is fixed. Like Fixed organization, TL is fixed in this case. The problem is to find an assignment of decision variables and a subcost functions to processors that minimize $max_i\ nt(i)$.

Mapping for Load Balancing

Given $G_Q = (V_Q, E_Q)$ and $G = (V, E)$, find
a decomposition $J(\tilde{x}) = \sum_i J^i$ and
a matching between $\{x_1, x_2, ..., x_M\}$ and V,
so that these minimize $max_i\ nt(i)$.

This problem is a hybrid of two problems; mapping strategy for parallel processing [11] and load balancing. Once a matching between $\{x_1, x_2, ..., x_M\}$ and V is determined, the problem is reduced to 'Load Balancing in a fixed structure' in section 3.1.1. Let us break up this problem into two parts; 1) matching, 2) decomposition of J. In matching part, if we assume $|V| = |V_Q|$, we can think of $|V_Q|!$ matchings. Because of 'Direct Communication' assumption, we have to elect from these matchings ones having the following property: if $(i, j) \in G_Q$, $\sigma(x_i)$ and $\sigma(x_j)$ are within two hops in G. If we run algorithms presented in section 3.1.1 for each of these matchings, and single out the one with smallest $min\ max_i\ nt(i)$, 'Mapping for Load Balancing' will be solved. Therefore, the focus should be on how to efficiently elect matchings with the property specified above. First, let us consider the question

of whether such a mapping exists.

Existence of a feasible Mapping

Given $G_Q = (V_Q, E_Q)$ and $G = (V, E)$, does there exist a one–to–one mapping

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

such that

$\sigma(x_i)$ and $\sigma(x_j)$ are within two hops from each other in G, $\forall (i,j) \in E_Q$.

Here is another way of viewing 'Existence of a feasible Mapping' in the framework of graph theory. Let us define a transformation of the graph $G = (V, E)$.

$$F(G) = (V, E_f)$$

where

$$E_f = E \cup \{(DM_i, DM_j) \notin E | DM_i \textit{ and } DM_j \textit{ are two hops from each other}\}$$

(See Figure 3-16 as an example) 'Existence of a feasible mapping' is equivalent to whether G_Q can be embedded in $F(G)$. If there is a one-to-one mapping σ in 'Existence of a feasible mapping' problem, each node i of G_Q can be embedded in $\sigma(x_i)$ in $F(G)$. For each pair $(i,j) \in E_Q$ $\sigma(x_i)$ and $\sigma(x_j)$ are within two hops from each other in G, so $(\sigma(x_i), \sigma(x_j)) \in F(G)$. Therefore, G_Q can be embedded in $F(G)$. Conversely, if G_Q is embedded in $F(G)$, we can construct a one-to-one mapping σ such that $\sigma(x_i)$ is the node in which $i \in V_Q$ is embedded. Then, for all edge $(i,j) \in G_Q$, $(\sigma(x_i), \sigma(x_j)) \in F(G)$ because G_Q is embedded in $F(G)$. Therefore, $\sigma(x_i)$ and $\sigma(x_j)$ are in two hops from each other in G.

The problem of determining whether G_Q can be embedded in $F(G)$ is a special case of 'Subgraph Isomorphism' problem. [9]

Subgraph Isomorphism

INSTANCE : Graphs $H_1 = (V_1, E_1)$, $H_2 = (V_2, E_2)$

QUESTION : Does H_1 contain a subgraph isomorphic to H_2?

If we restrict H_1 to be a set of graph

$$\{graph\ H \mid \exists\ a\ graph\ \Gamma\ such\ that\ F(\Gamma) = H\}$$

this restricted version is equivalent to 'Existence of a feasible Mapping'. (Note that some graphs do not belong to this set. For an example, see the Appendix.)

Though 'Existence of a feasible Mapping' is equivalent to a subproblem of 'Subgraph Isomorphism', 'Existence of a feasible Mapping' is very similar in structure to the original 'Subgraph Isomorphism', which is known to be NP–complete. Therefore, we conjecture that this problem is NP–complete. Consequently, we conjecture that 'Mapping for Load Balancing' is also NP–complete.

3.2 RL as an amount of communication

When RL is used as a measure of the amount of communication, task allocation in a flexible organization becomes tractable. Also, it turns out that task allocation in a flexible organization is a special case of task allocation in a fixed organization. Therefore, the case of flexible organization will be presented first, and the idea used for this case will be generalized for the case of fixed organizational structure.

3.2.1 Flexible organizational structure

Minimal Superposed Link

Given $J = x^T Q x$, nt_i^*, find a decomposition that minimizes RL such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

If communication load is defined by RL (section 2.2.3), minimization of communication load is done by a special binary integer linear programming. Let

$$V = \{DM_1, DM_2, DM_3, ..., DM_M\}$$

$$E_Q = \{unordered\ pair\ (i,j) | the\ cross\ term\ x_i x_j\ is\ in\ J\ \}$$

Let the variable

$$X_{ijk} = \begin{cases} 1 & \text{if } x_i x_j \text{ is assigned to } DM_k \\ 0 & \text{otherwise} \end{cases}$$

Since each cross term is assigned to one processor,

$$\sum_{k=1}^{M} X_{ijk} = 1$$

Let the capacity of processor DM_i be nt_i^* (the number of cross terms it can handle). The number of cross terms assigned to DM_k must be within its capacity, so

$$\sum_{(i,j)\in A} X_{ijk} \leq nt_i^*$$

For a variable $X_{ijk} = 1$, if k is i or j, one link is introduced; namely a link between DM_i and DM_j. If k is neither i nor j, two links are introduced; namely a link between DM_i and DM_k and a link between DM_j and DM_k. (section 2.2.3) Therefore, communication load, RL is

$$\sum_{k\neq i, k\neq j} 2X_{ijk} + \sum_{k=i} X_{ijk} + \sum_{k=j} X_{ijk}$$

In summary the decomposition problem is reduced to the following binary integer linear programming problem. We call this problem binary integer linear programming problem 'BIL'.

minimize $\sum_{k\neq i,k\neq j} 2X_{ijk} + \sum_{k=i} X_{ijk} + \sum_{k=j} X_{ijk}$

such that

$\sum_{k=1}^{M} X_{ijk} = 1 \qquad \forall (i,j) \in E_Q$

$\sum_{(i,j)\in E_Q} X_{ijk} \quad \leq nt_k^*.$ .....(BIL)

$X_{ijk} \ = 0 \ for \ (i,j) \notin E_Q$

$X_{ijk} \in \{0,1\}$

Without the last constraint, which is the integrality constraint, this linear programming problem is a linear network problem. In fact this problem can be formulated as a 'minimum cost network flow' [6] problem. Figure 3-17 shows a minimum cost network flow problem that is equivalent to BIL. Each node, m_{ij} corresponds to a pair $(i,j) \in E_Q$. Each node, n_i corresponds to $DM_i \in V$. Minimum cost network flow problems can be easily (polynomially) transformed to Hitchcock problems.[6] Figure 3-18 shows how to transform our minimum cost flow problem to a Hitchcock problem. In Hitchcock problem a feasible set is represented by a system of linear equations, and the coefficients of those linear equations form a node–arc incidence matrix. A node–arc incidence matrix is 'totally unimodular' [6]. Therefore, optimal vertices have integer elements. Consequently, a simplex algorithm produces an integer optimal flow of the network, so a simplex algorithm solves BIL. We can expect any variation of simplex algorithm to solve BIL as long as feasible points move from vertex to vertex. We now explicitly show that some well–known algorithms can be applied to to solve BIL.

Primal–Dual algorithm

Primal–Dual algorithm [6] can be run to solve the minimum cost network flow problem corresponding to BIL (e.g. Figure 3-17). Let us call the network N. Let $\tilde{f}$ be a vector that indicates the flow of the network, and let $f(i,j)$ be a flow through a directed arc (i,j). For a feasible flow, $\tilde{f}$, Primal–Dual algorithm [6] constructs an increment network, $N'(\tilde{f})$, for the current feasible flow by changing arcs of N as follows: for each directed arc (m_{ij},k) with cost c, add a directed arc (k,m_{ij})

with capacity $[0, f(m_{ij}, k)]$. and with cost $-c$. (Figure 3-19a) For each arc (n_i, t), change the capacity to $[0,\ nt_i^* - f(n_i, t)]$ and add a directed arc (t, n_i) with a capacity $[0,\ f(n_i, t)]$. (Figure 3-19b) At each iteration, Primal–Dual algorithm finds a negative–cost cycle in N' and increases flow through arcs of this cycle by the minimum over the upper bounds of arcs in the negative cycle.

We claim that if we choose an integer initial flow, the algorithm will produce an integer optimal flow.

Proof

Let $\tilde{f}$ be a current flow that is integral. Since the bound of flow along any arc of the network (e.g. Figure 3-17) is integer, a bound of flow along any arc in the increment network $N'(\tilde{f})$ is integer. Therefore, if a negative cycle exists in $N'(\tilde{f})$, the incremental flow $\bar{f}$ is integer. This incremental flow is along the arcs of that negative cycle. For the arcs that is not in the negative cycle, their flow does not change. Therefore, the new flow is also integer. By induction, all the flows in sequence are integers.

The integer flows of minimum cost network flow problem correspond to feasible assignments of cross terms. The cost of flow corresponds to the communication load required by the assignments of cross terms. Therefore, if we manage to have a feasible, integer initial flow, we can obtain optimal decomposition by running Primal–Dual algorithm for the network exemplified by Figure 3-17. Now the focus of discussion should be on how to obtain a feasible, integer initial flow. We can obtain a feasible, integer initial flow by converting the network like Figure 3-17 into a single–source, single–sink network. Add a source node, s to the network like Figure 3-17 and construct a link from s to each node m_{ij}. Let the capacity of these links be $[0, 1]$ (Figure 3-20) By running max-flow algorithm like Algorithm 3.1 on this

single–source, single–sink network, we can obtain integer initial flow through arcs of type (m_{ij}, n_k) and (n_k, t).

Load Balancing with Limited Superposed Link

Given $J = x^T Q x$, RL^*, find a decomposition that minimizes $max_i\ nt(i)$ such that $RL \leq RL^*$.

The idea used for 'Load Balancing in a Fixed Structure' (section 3.1.1) can be used for this problem. The smallest $max_i\ nt(i)$ we can possibly have is $\lceil \frac{|E_Q|}{M} \rceil$, where $|E_Q|$ is total number of cross terms in J, and M is the number of processors. For the most unbalanced decomposition, $max_i\ nt(i)$ is $|E_Q|$ (if the cross terms are assigned to a single processor). Initially, we set the capacity of all the arcs of type (n_k, t) to be $\lceil \frac{|E_Q|}{M} \rceil$, and minimize RL. Minimizing RL is solving 'Minimal Superposed Link', so we can use Primal–Dual algorithm. If we obtain $RL \leq RL^*$, we achieve $min\ max_i nt(i)$; otherwise, increase the capacity of arcs (n_k, t) by one. If we do not obtain $RL \leq RL^*$ until the capacity of these arcs becomes $|E_Q|$, given instance of the problem is infeasible. In other words, no matter how we decompose J, we need more than RL^* for amount of communication. The following algorithm summarizes our discussion:

Algorithm 3.4

1. Construct a network corresponding to BIL, where $nt_k^* = dummy$ for $k = 1, 2, .., M$
2. Set $dummy = \lceil \frac{|E_Q|}{M} \rceil$

3. Do while $dummy \leq |E_Q|$

   Run Primal–Dual algorithm

   If $RL \leq RL^*$, $min\ max_i\ nt(i) = dummy$; terminate

   $dummy := dummy + 1$

4. Return "Infeasible"

3.2.2 Fixed organization

In previous sections the amount of communication was defined as the total number of message transfers, where a processor must transmit one message for each cross term that is assigned to another processor, and which involves its variable. A flexible organizational structure was assumed. It means that the task allocator has an authority to construct a link between any processors if necessary. If one or more messages need to be transferred in the previous section, the allocator can freely put links between those processors. In this section, the case of fixed organizational structure is discussed. The links between processors are fixed; therefore, the allocator or the allocation algorithm must decompose the global cost function such that the message might be transferred only through existing links. Under this constraint we want to minimize the amount of necessary communication defined by RL.

Minimal Superposed Link in a Fixed Structure

Given a graph G, G_Q and nt_i^*, find a decoposition that minimizes the RL such that the message is only transferred through the edges of G and such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

When an organization has fixed structure a priori, (here, structure simply means which subdivision can communicate with which.) and the task allocator should partition the global task, he must partition it in a way that the necessary message transfer among subtasks should be allowed only those pairs of subdivisions that can communicate with each other. Each subdivision has its own capacity. Consider a situation where the leader of this organization wants to allocate the tasks with lowest amount of message transfers possible. The formulation in this section mathematically models this task allocation problem of an organization. We can also apply this problem to a situation of a computer processor network on which distributed algorithm is running.

This problem is a generalization of the problem solved in section 3.2.1. sections. If the fixed structure of this problem is a complete graph, this problem is exactly equivalent to the problem solved in section 3.2.1. We can design a polynomial–time algorithm by slightly modifying the algorithm introduced in the previous sections. Instead of running 'Primal–Dual algorithm' on the network in Figure 3-17, we can run this algorithm on a modified network that corresponds to the fixed organizational structure. The following algorithm shows how to construct such a network.

Algorithm 3.5

Phase 1

1. Create $|E_Q|$ nodes corresponding to cross terms of J. Let m_{ij} denote the node corresponding to $x_i x_j$.
2. Create M nodes corresponding to processors. Let n_i denote the node corresponding to DM_i.
3. For each cross term $x_i x_j$, do

If $(DM_i, DM_j) \in E$, make an arc from m_{ij} to n_i, and make an arc from m_{ij} to n_j. Let the cost of these two arcs be 1.

For each neighbor of DM_i in G, say $DM_k \neq DM_j$,

If $(DM_k, DM_j) \in E$, make an arc from m_{ij} to n_k. Let the cost of this arc be 1.

If no arc is made for $x_i x_j$, terminate with an output;

"Organization cannot handle this task."

All the arcs created in Step 3 have infinite capacity.

4. Create the sink node t and make an arc from each n_i to t. Each arc (n_i, t) created at Step 4 has capacity nt_i^*, and cost 0.

At Step 3, if no arc is made for some cross term $x_i x_j$, that means DM_i and DM_j are not within two hops from each other. Therefore, this term cannot be assigned to any processor. (Lemma 2-1) The organization cannot handle this task.

Phase 2

Run the 'Primal–Dual algorithm' as in the previous section.

We can also consider $max_i nt(i)$ as our objective while setting the amount of communication defined by RL as a constraint.

Load Balancing with Limited Superposed Links in a Fixed Structure

Given a graph G, G_Q, and RL^*, find a decomposition that minimizes $max_i\ nt(i)$ such that the message is only transferred through the edges of G, and $RL \leq RL^*$.

The same idea used for 'Load Balancing with Limited Superposed Link' can

be used. As in Algorithm 3.5, which is for 'Minimal Superposed Link in a Fixed Structure', a graph is constructed corresponding to the problem instance. The only difference is that the capacity of all the arcs of the type (n_i, t) are set to be equal; initially $\lceil \frac{|E_Q|}{M} \rceil$, the best balance possible. At each iteration, we minimize RL using Phase 2 of Algorithm 3.5. If we obtain $RL \leq RL^*$, we have achieved $min\ max_i nt(i)$. Otherwise increase the capacity of all the arcs of the type (n_i, t) by one.

Algorithm 3.6

1. Set $dummy = \lceil \frac{|E_Q|}{M} \rceil$
2. Run Step 1 throug Step 3 of Phase 1 of Algorithm 3.5
3. Create the sink node t and make an arc from each n_i to t. Each arc (n_i, t) created at Step 4 has capacity $dummy$, and cost 0.
4. Do while $dummy \leq |E_Q|$

   Run Phase 2 of Algorithm 3.5

   If $RL \leq RL^*$, $min\ max_i\ nt(i) = dummy$; terminate

   $dummy := dummy + 1$
5. Return "Infeasible"

3.2.3 Semi-flexible organizational structure

In this section we will discuss the situation where the communication structure of an organization is fixed, but the task allocator has the freedom to assign to subdivisions decision variables as well as subtasks. This situation is best explained by the following mathematical statement of the formulation.

Mapping for Minimal Superposed Link

Given G_Q, $G = (V, E)$, and nt_i^* for $i = 1, 2, .., M$, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize RL such that $nt(i) \leq nt_i^*$, $i = 1, 2, .., M$.

Mapping for Load Balancing with Limited Superposed Link

Given G_Q, $G = (V, E)$, and RL^*, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize $max_i nt(i)$ such that $RL \leq RL^*$.

The one-to-one mapping σ mathematically represents the assignment of decision variables to subdivisions. Each variable, x_i represents the decision which each subdivision is delegated to make, where the global task is to make a decision $(x_1, x_2, ..., x_M)$ that minimizes the cost function J.

Here is the recognition version of these two problems

Problem 1

Given $G_Q = (V_Q, E_Q)$, $G = (V, E)$, and nt_i^* for $i = 1, 2, .., M$, RL^*,
does there exist a one-to-one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition such that

$$nt(i) \leq nt_i^*, \quad i = 1, 2, .., M, \qquad RL \leq RL^* \quad ?$$

Theorem 3-14

The recognition version of two problems above (Problem 1) is NP–complete.

Proof

Let us restrict the problem by making

$$nt_i^* = |E_Q| \qquad \forall i$$

and

$$RL^* = |E_Q|$$

This restricted version is, then, equivalent to 'subgraph isomorphism' [9] problem. Now, the restricted version is

Problem 2

INSTANCE: $G_Q = (V_Q, E_Q)$, $G = (V, E)$,

PROBLEM : $nt(i) = |E_Q| \quad \forall i$;

does there exist a one-to-one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition such that

$nt(i) \leq nt_i^*, \quad i = 1, 2, .., M$ and $\qquad RL \leq |E_Q|$?

Subgraph Isomorphism

INSTANCE: $G_Q = (V_Q, E_Q)$, $G = (V, E)$,

PROBLEM : Is G_Q a subgraph of G

Suppose the instance of Problem 2 is 'yes'. For any one-to-one mapping σ, the assignment of one cross term increases RL either by 1 or 2. Since we have $|E_Q|$ cross terms, and $RL \leq |E_Q|$, RL must be increased only by 1 for the assignment of each cross term $x_i x_j$. Therefore, there must exist a link between the node $\sigma(x_i)$ and $\sigma(x_j)$. (Otherwise, the assignment of $x_i x_j$ increases RL by 2.) Therefore, G_Q can be embedded in G.

Suppose the instance of 'Subgraph Isomorphism' is 'yes'. Take the subgraph of G, which is isomorphic to G_Q. We can define the mapping such that $\sigma(x_i)$ is the node of G corresponding in this isomorphism to the node of G_Q representing x_i. As long as the decomposition is concerned, assign the cross term $x_i x_j$ either to $\sigma(x_i)$ or to $\sigma(x_j)$. This way, the assignment of each cross term ncreases RL only by 1. Therefore, RL is $|E_Q|$ in the end. Since the number of cross terms are $|E_Q|$, $nt(i) \leq |E_Q|$ for any decomposition. Thus, the instance of Problem 2 is 'yes'.

We have shown that subgraph isomorphism problem is polynomially transformed to Problem 2. Subgraph isomorphism problem is known to be NP–complete. It is obvious that Problem 2 is in NP. Therefore, Problem 2 is NP–complete.

Let us consider a special case of our problem; namely,

$$|V| = |V_Q|$$

If we restrict the general graph isomorphism problem to a special case where $|V| = |V_Q|$, this special graph isomorphism problem is still NP–complete. (If we again

restrict to an instance where G_Q is a ring, it is equivalent to Hamiltonian circuit problem.) Therefore, the following, special case of our problems are still NP–complete.

Problem 3

INSTANCE: $G_Q = (V_Q, E_Q)$, $G = (V, E)$, $|V| = |V_Q|$

PROBLEM : $nt(i) = |E_Q| \ \forall i$;

does there exist a one-to-one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition such that

$nt(i) \leq nt_i^*, \quad i = 1, 2, .., M$ and $\qquad RL \leq |E_Q|$?

Problem 4

INSTANCE: $G_Q = (V_Q, E_Q)$ is a ring, $G = (V, E)$,$|V| = |V_Q|$

PROBLEM : $nt(i) = |E_Q| \ \forall i$; $\qquad$ does there exist a one-to-one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition such that

$nt(i) \leq nt_i^*, \quad i = 1, 2, .., M$ and $\qquad RL \leq |E_Q|$?

Figures of Chapter 3

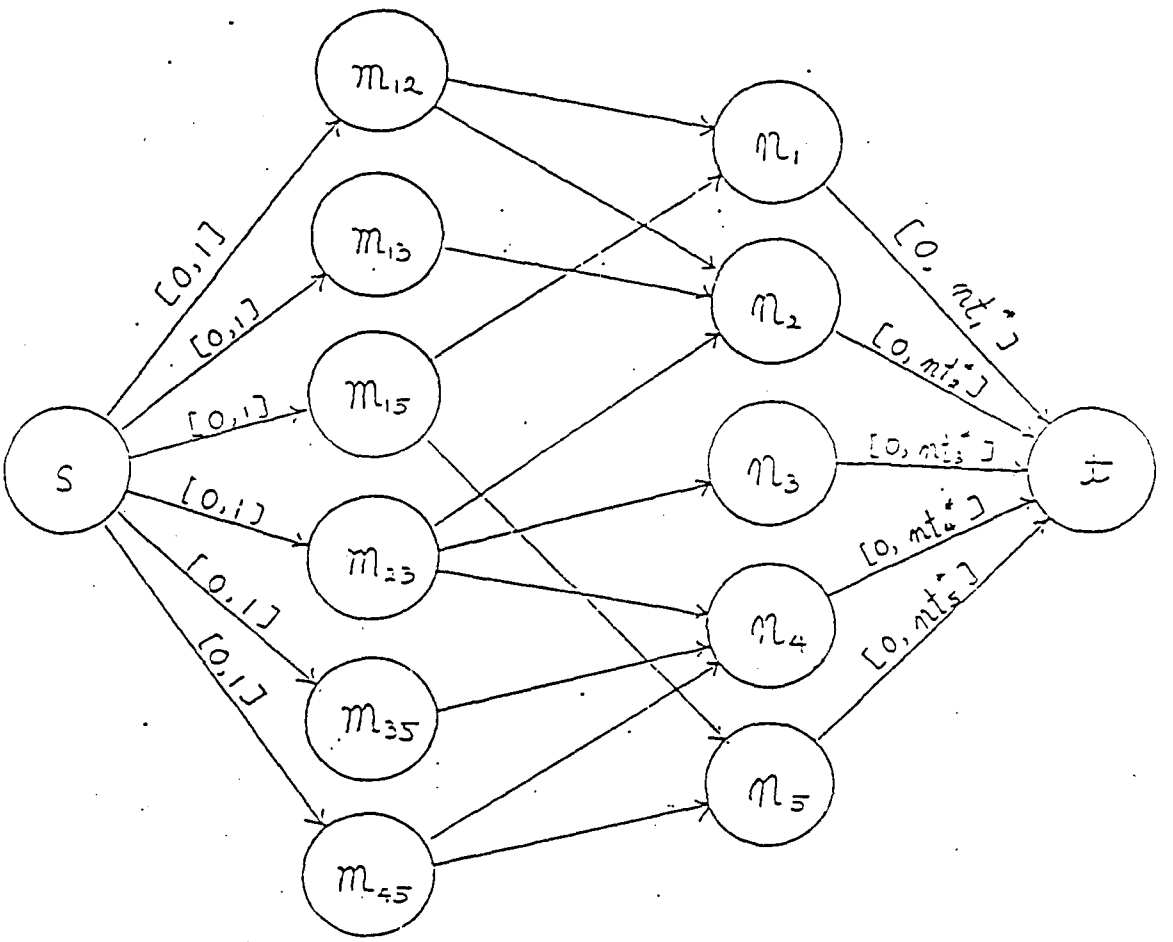
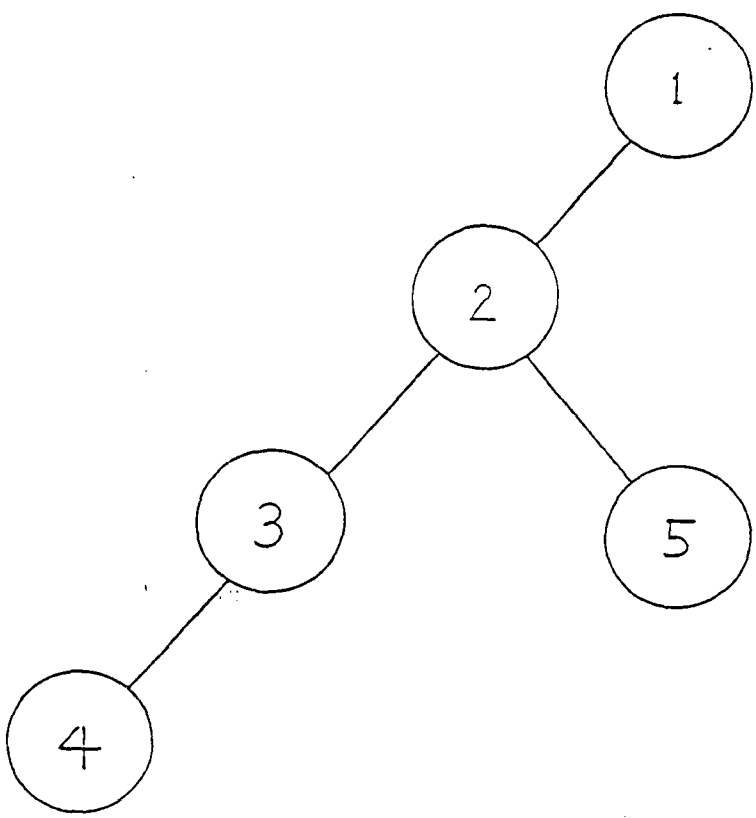


Figure 3-1 Digraph constructed by Phase 1 of Algorithm 3.1

Example

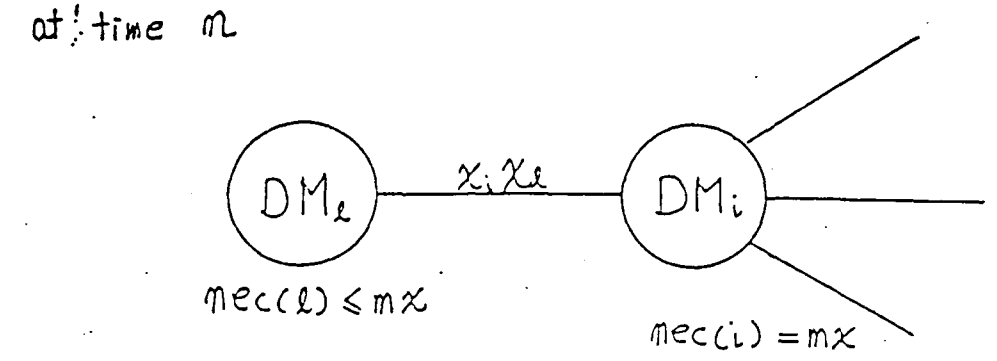
$$J(x_1, x_2, x_3, x_4, x_5) = 10x_1^2 + 10x_2^2 + 10x_3^2 + 10x_4^2 + 10x_5^2 + x_1x_2 + x_1x_4 + x_1x_5 + x_3x_4$$

$T = (V, E_T)$, a fixed tree

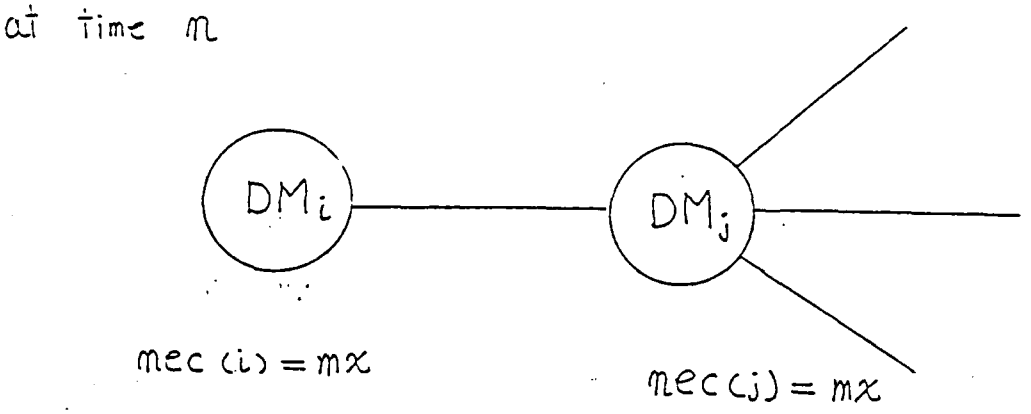


J has a cross term x_1x_4, but DM_1 and DM_4 are three hops apart.

Figure 3-2 Illustration for Algorithm 3.3

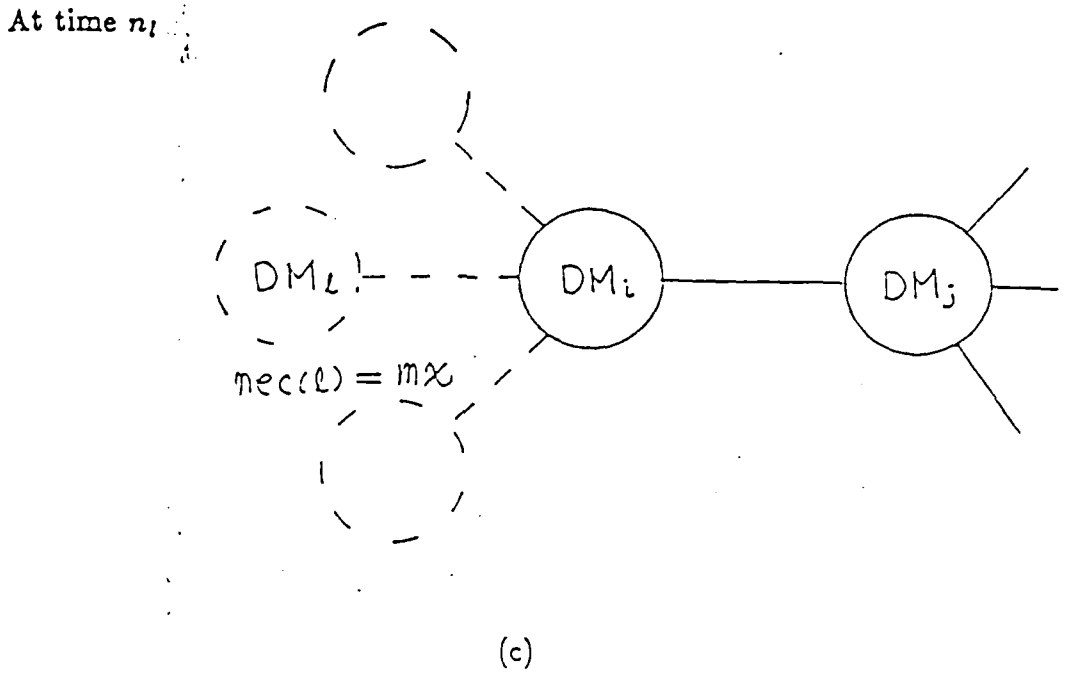


(a)

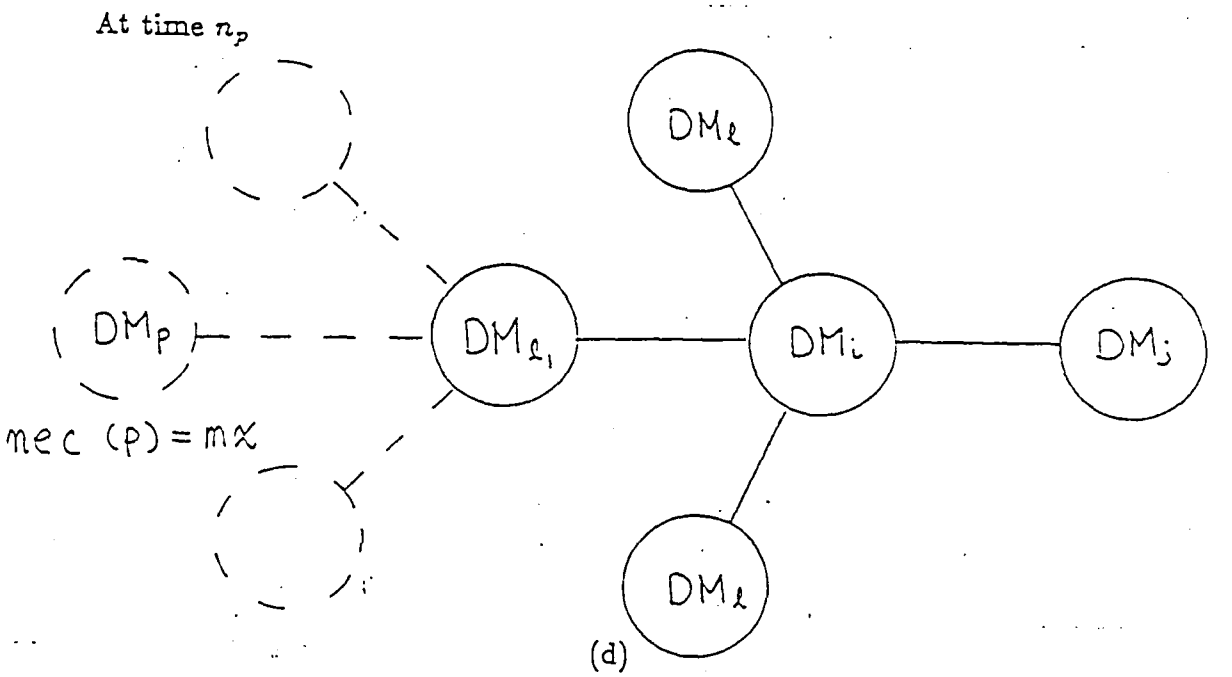


(b)

Figure 3-3 (a) (b) Algorithm 3.3 for Fixed Tree



(c)



(d)

Figure 3-3 (c) (d) Algorithm 3.3 for Fixed Tree

$$x_i x_j \quad |i-j| < W$$

Without loss of generality, assume $j > i$.

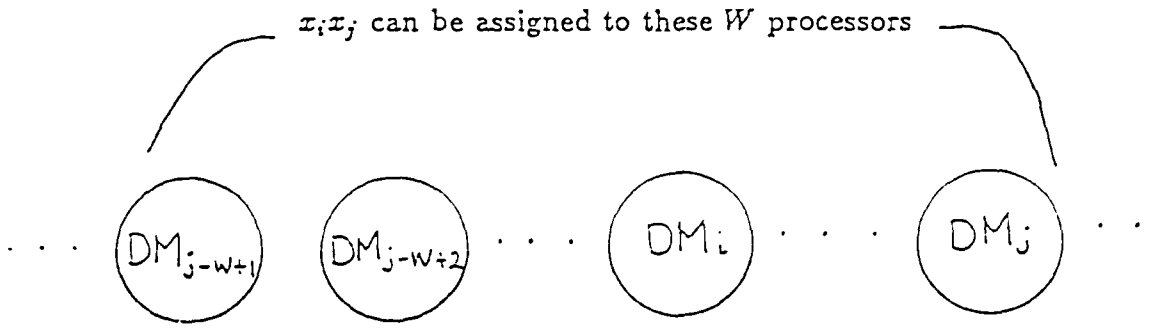
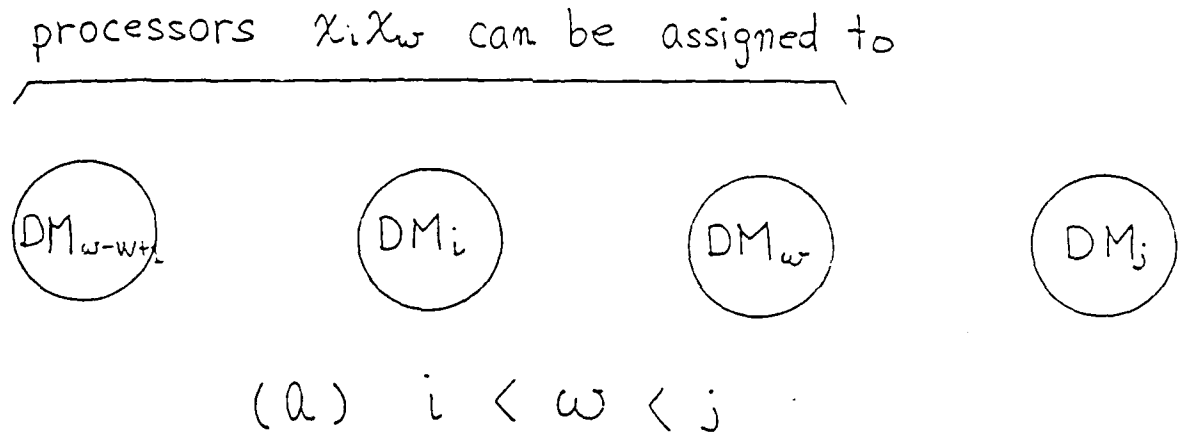


Figure 3-4 Constraint 2 of Minimal Link for bandwidth limited cost function



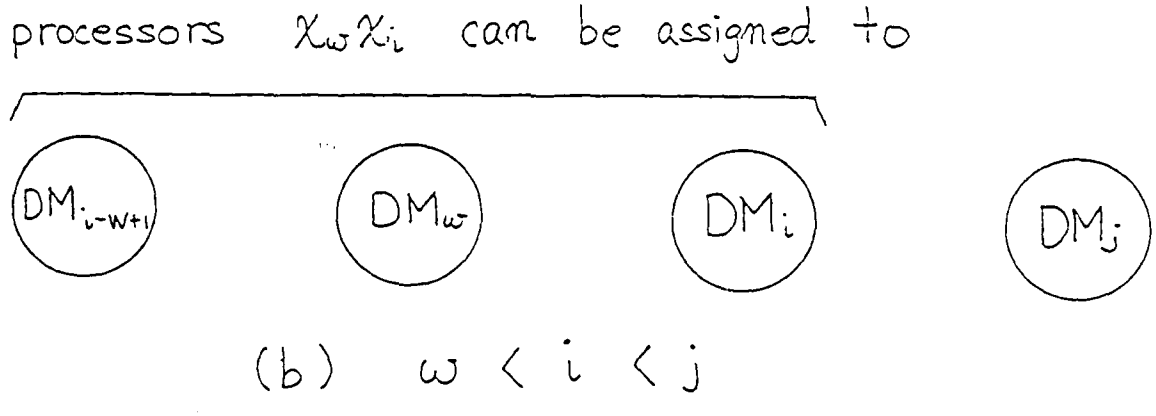


Figure 3-5 Locality of interaction

At the beginning of step t

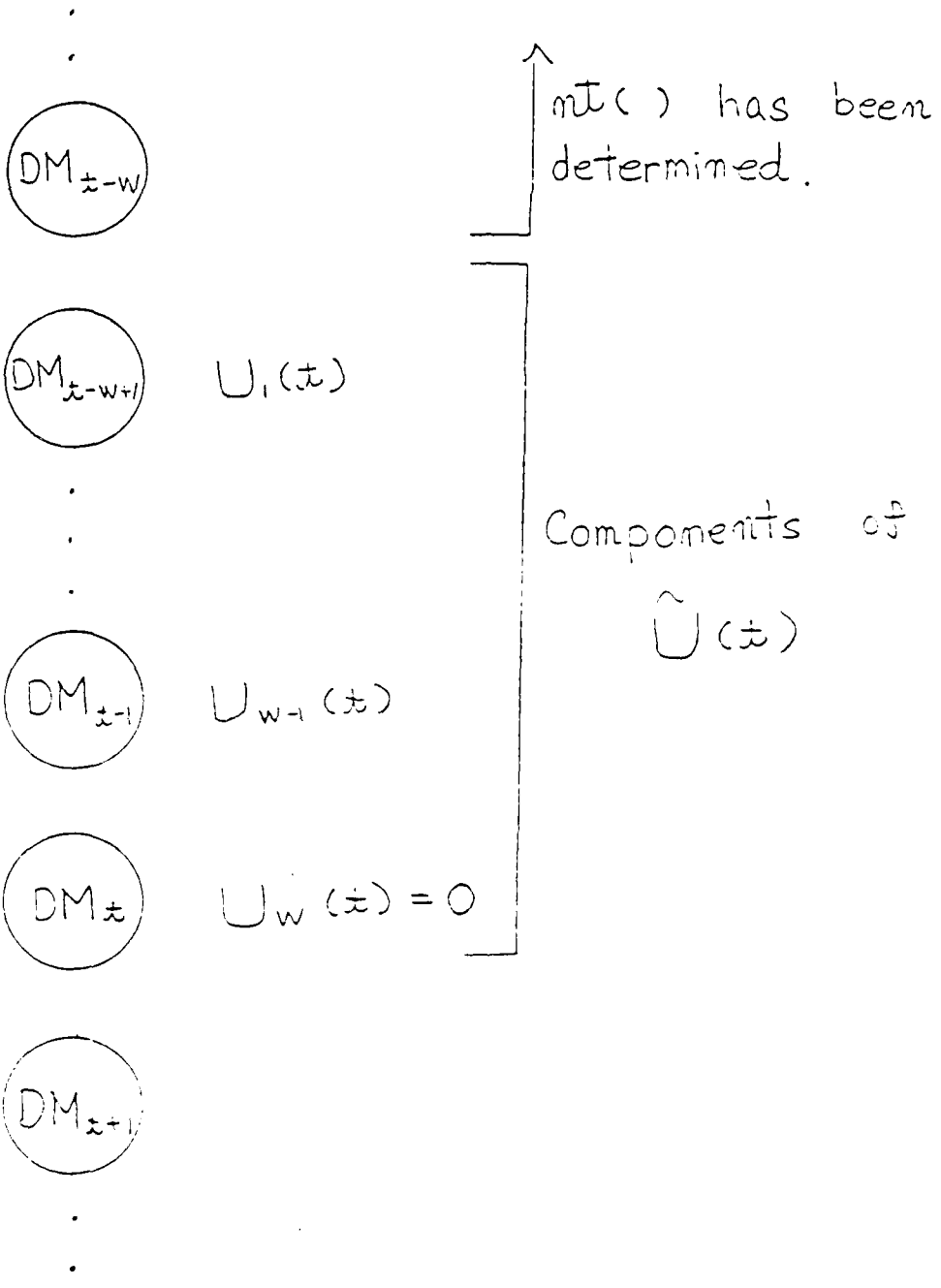
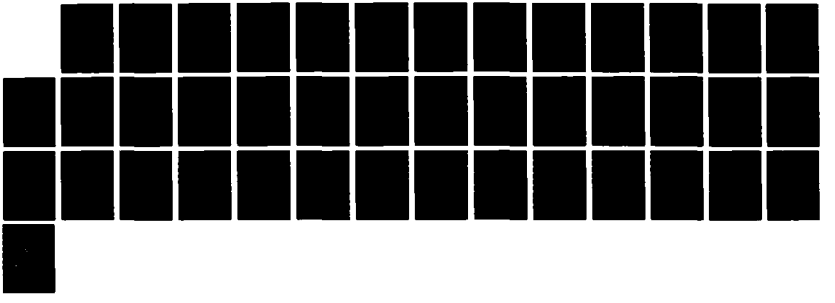


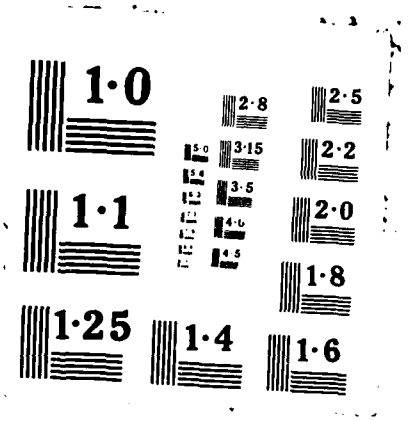
Figure 3-6 State vector $\tilde{U}(t)$

AD-A188 699 TASK ALLOCATION FOR EFFICIENT PERFORMANCE OF A DECENTRALIZED ORGANIZATION(U) MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION AND D.. C LEE

2/2

UNCLASSIFIED SEP 87 LIDS-TH-1706 N00014-85-K-0519 F/G 5/3 NL




1·0
2·8
2·5
3·15
2·2
1·1
3·5
2·0
1·8
1·25
1·4
1·6

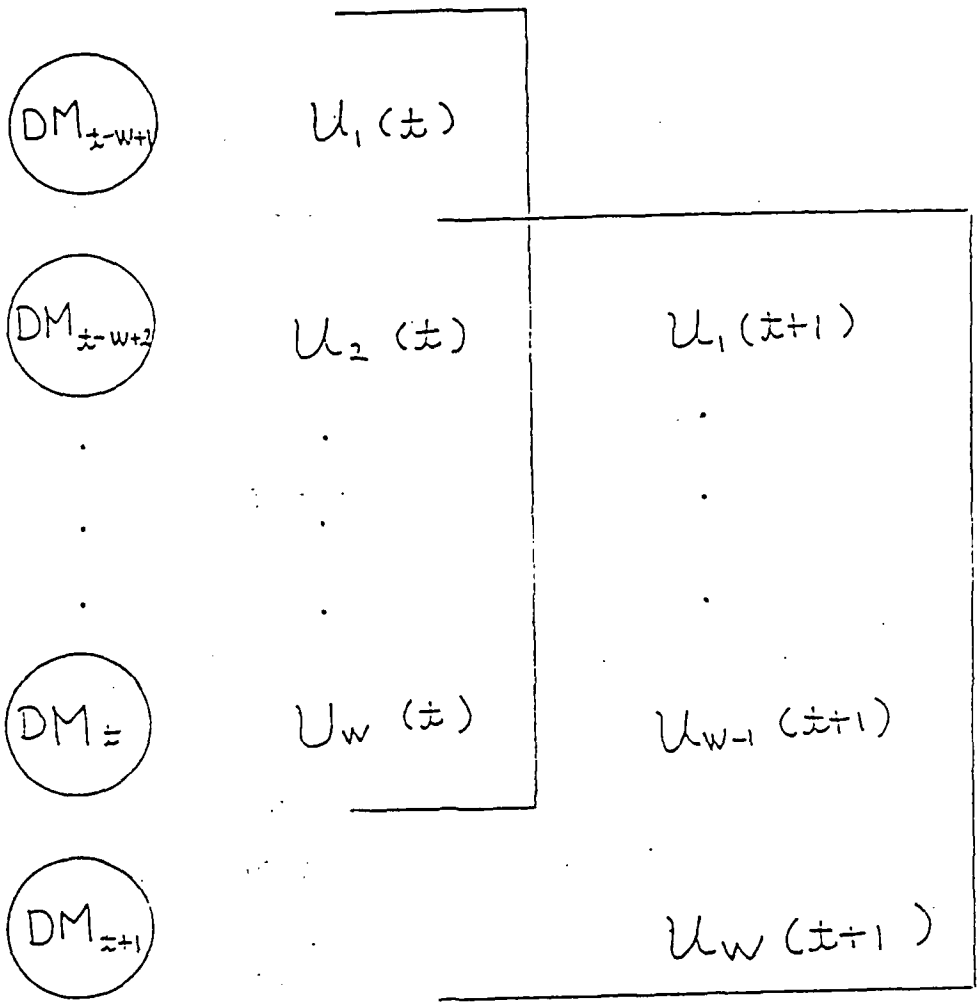


Figure 3-7 Evolution of the state vector $\tilde{U}$

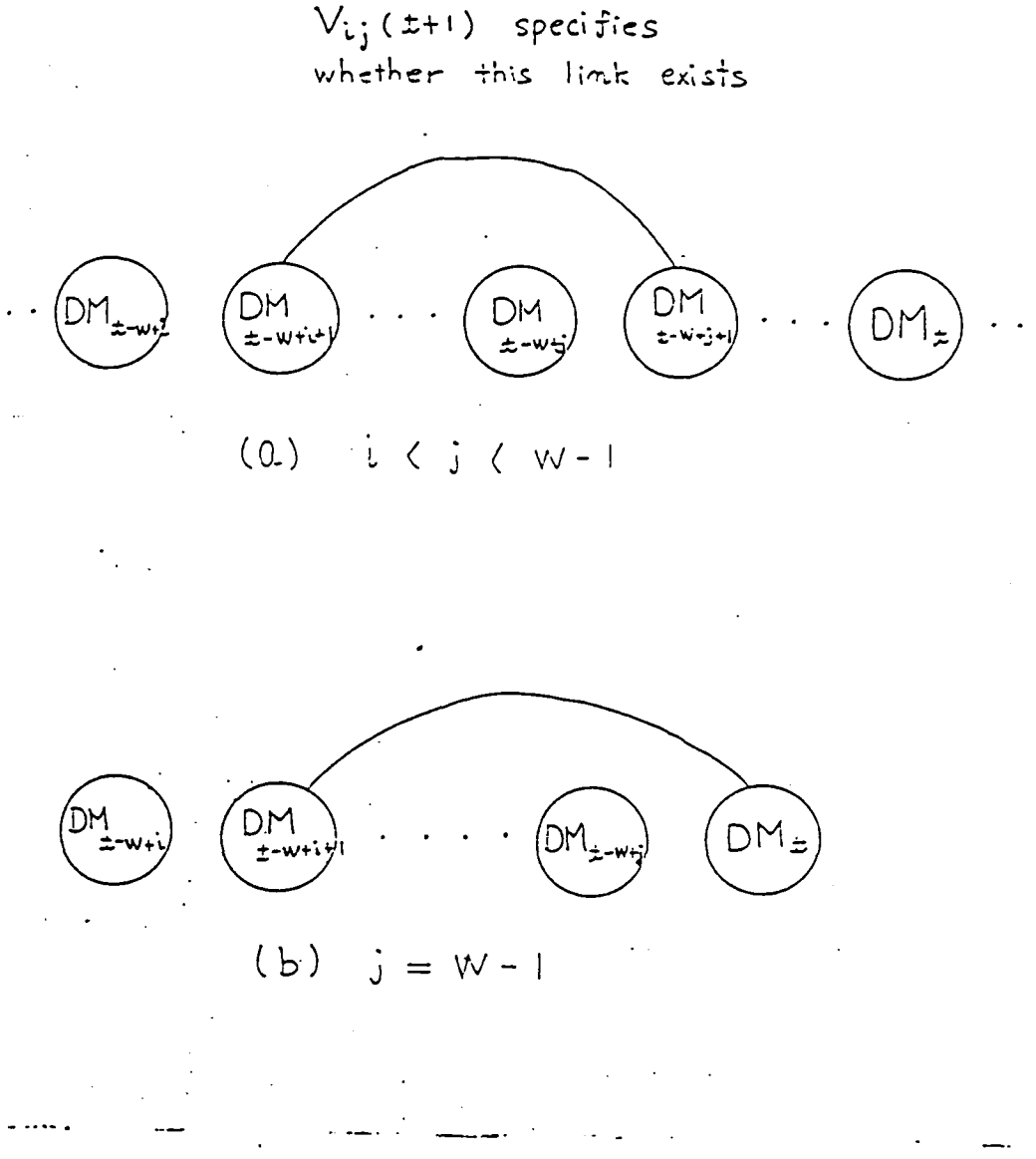
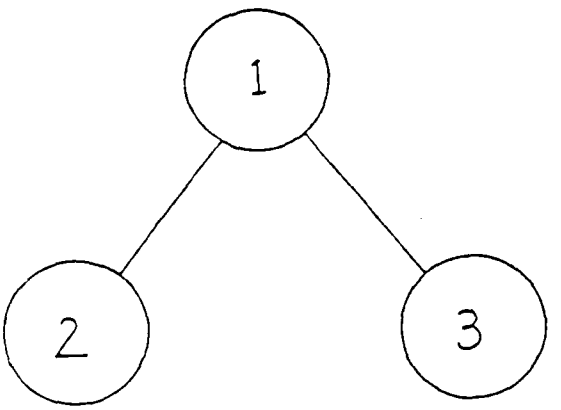


Figure 3-8 State variable V_{ij}

Example

$$J(x_1, x_2, x_3) = 10x_1^2 + 10x_2^2 + 10x_3^2 + x_1x_2 + x_1x_3$$

G_Q



$G = (V, E)$

$max_i\ nt(i) = 1$

$min\ max_i\ nt(i) = 1$

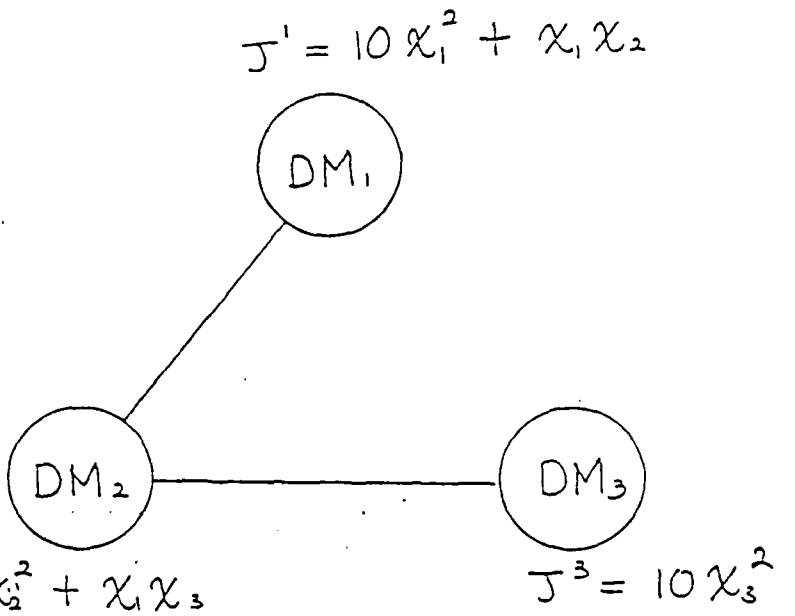


Figure 3-9 $max_i\ nt(i)$ is minimized in a tree that is not a subset of G_Q

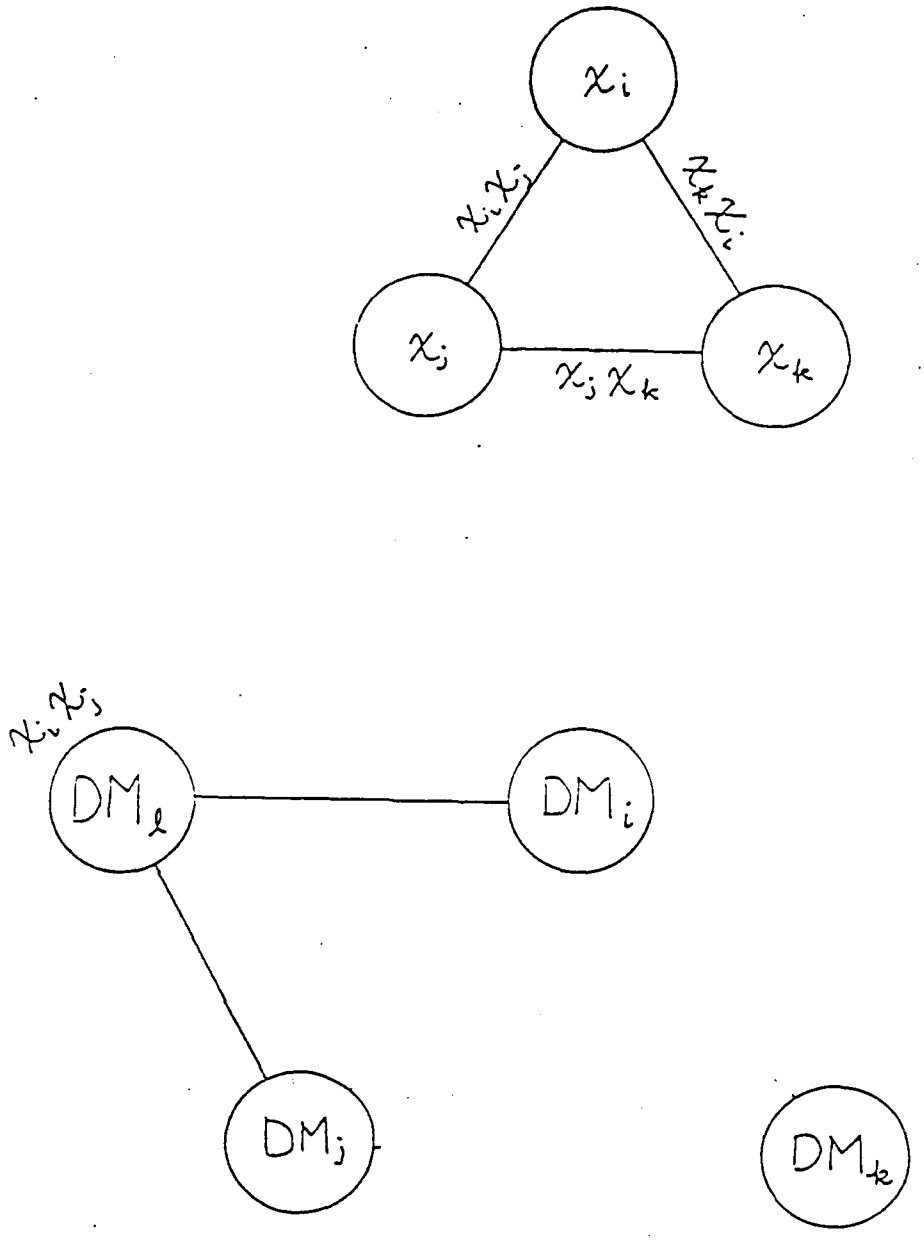


Figure 3-10 a: The condition in Lemma 3-7

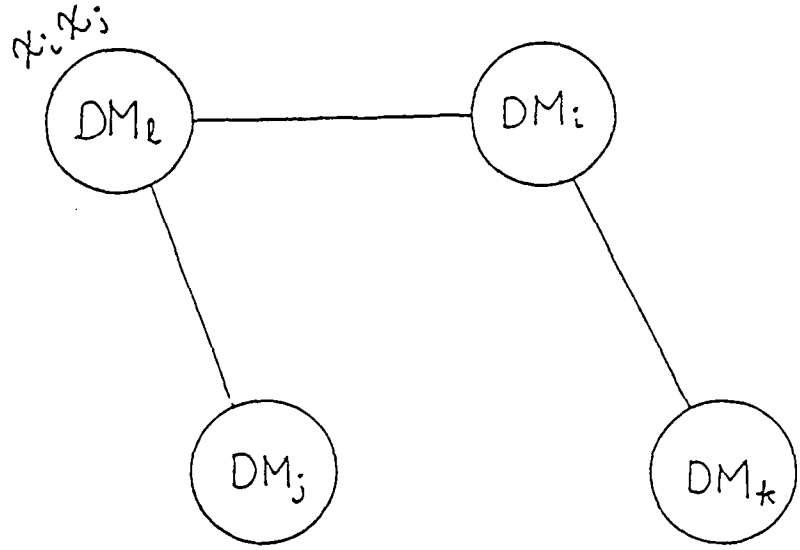


Figure 3-10 b: Illustration of the proof of Lemma 3-7

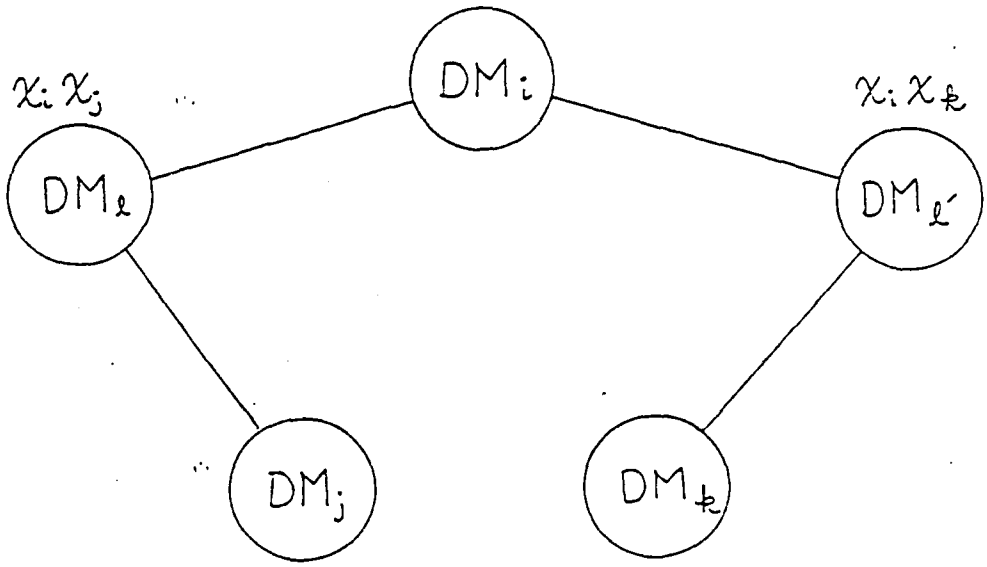


Figure 3-10 c: Illustration of the proof of Lemma 3-7

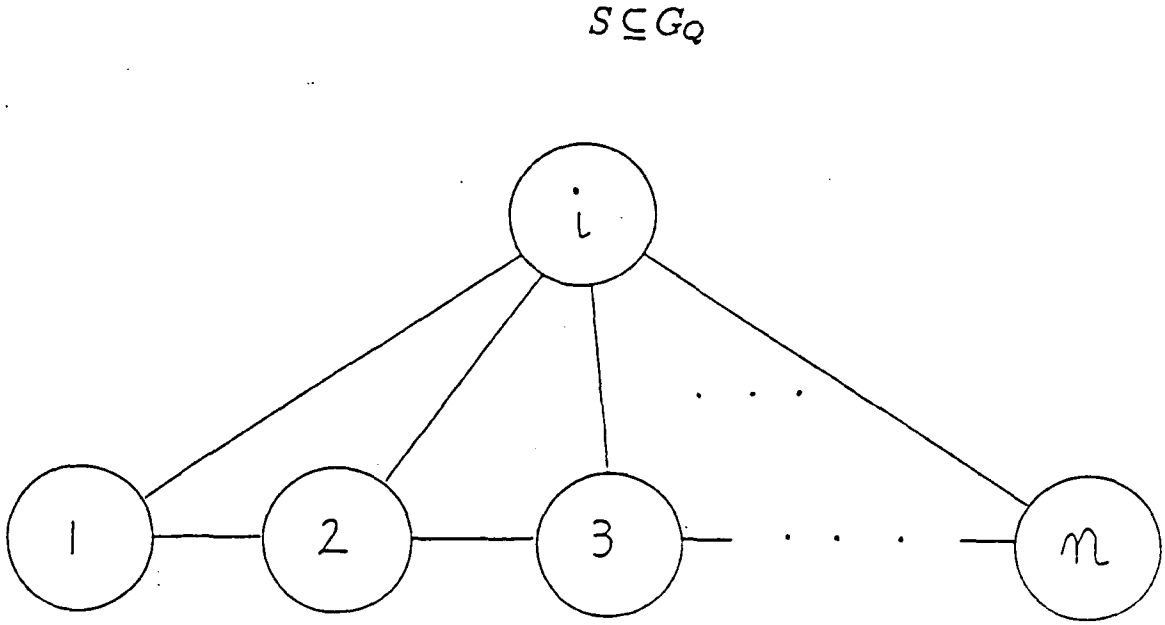


Figure 3-11 Subgraph S described by eqn (3-1), (3-2)

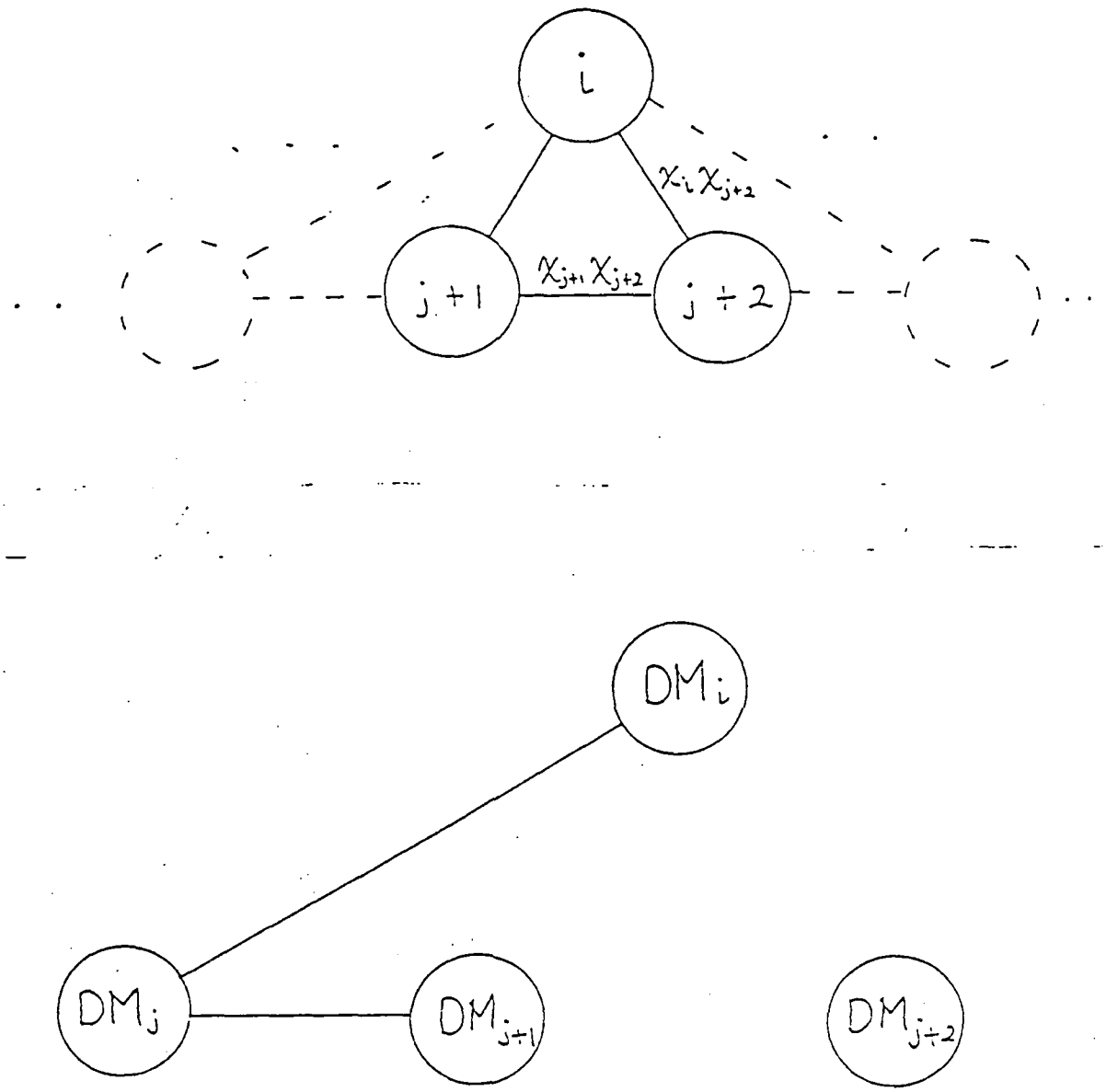


Figure 3-12 Illustration of the proof of Theorem 3-10

$$G_Q = (V_Q, E_Q)$$

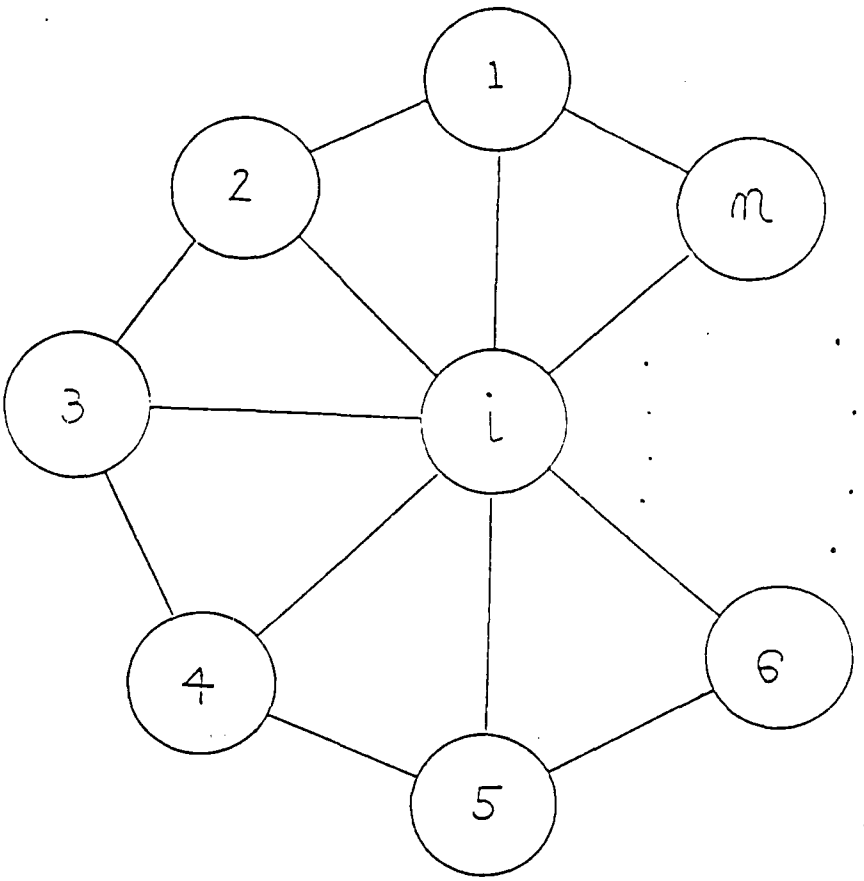
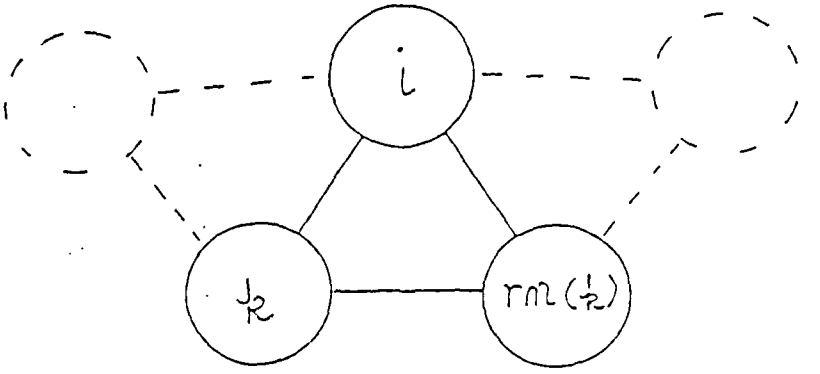


Figure 3-13 G_Q described by eqn (3-3), (3-4)



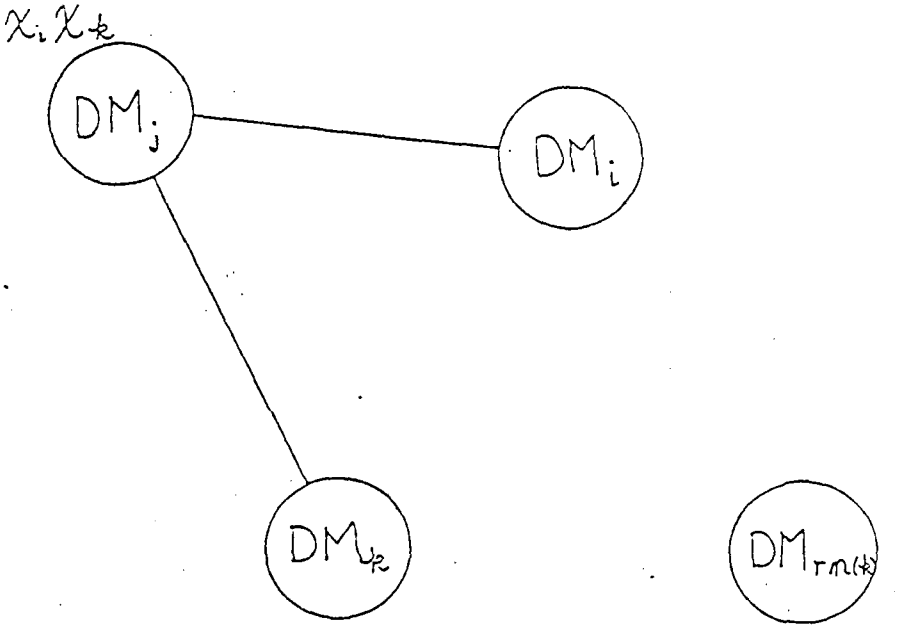


Figure 3-14 Illustration of the proof of Theorem 3-11

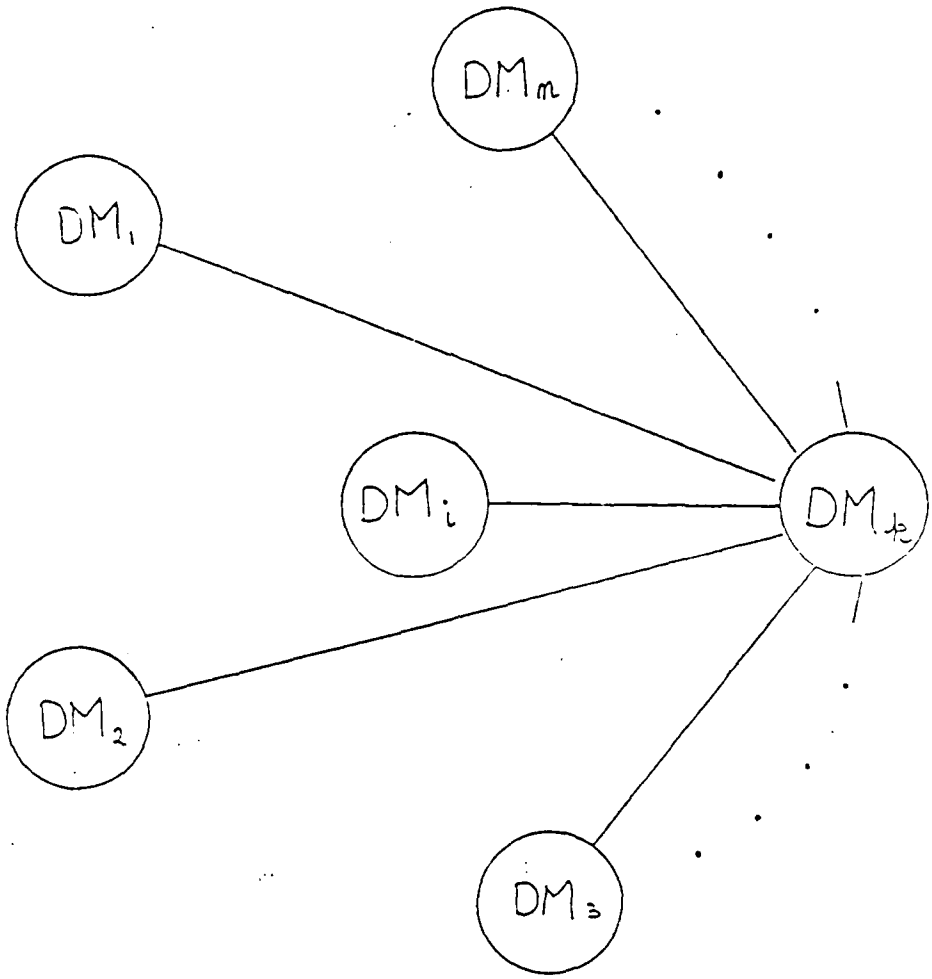


Figure 3-15 a : Illustration of the proof of Theorem 3-13

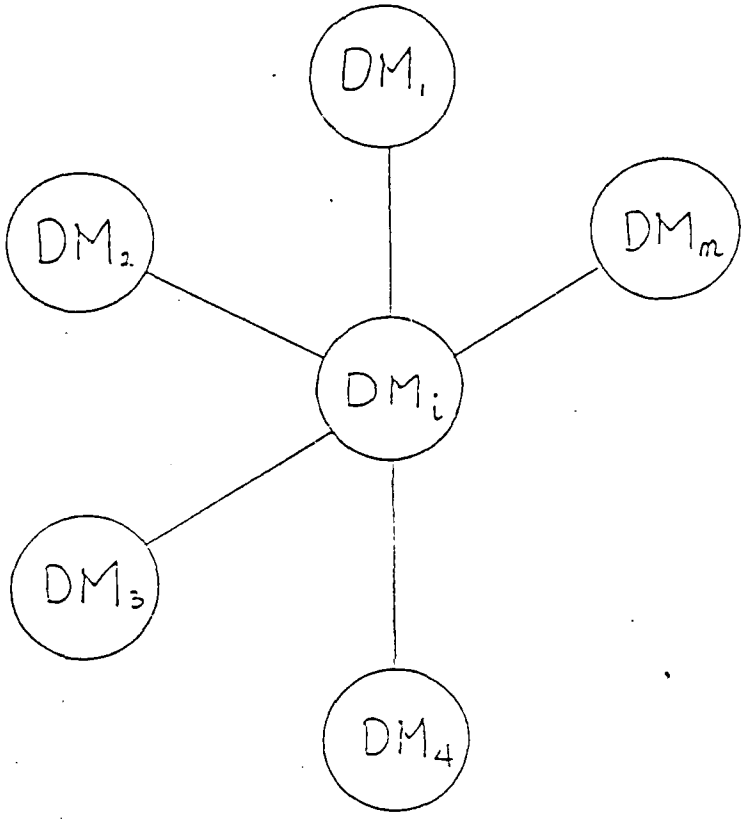


Figure 3-15 b : Illustration of the proof of Theorem 3-13

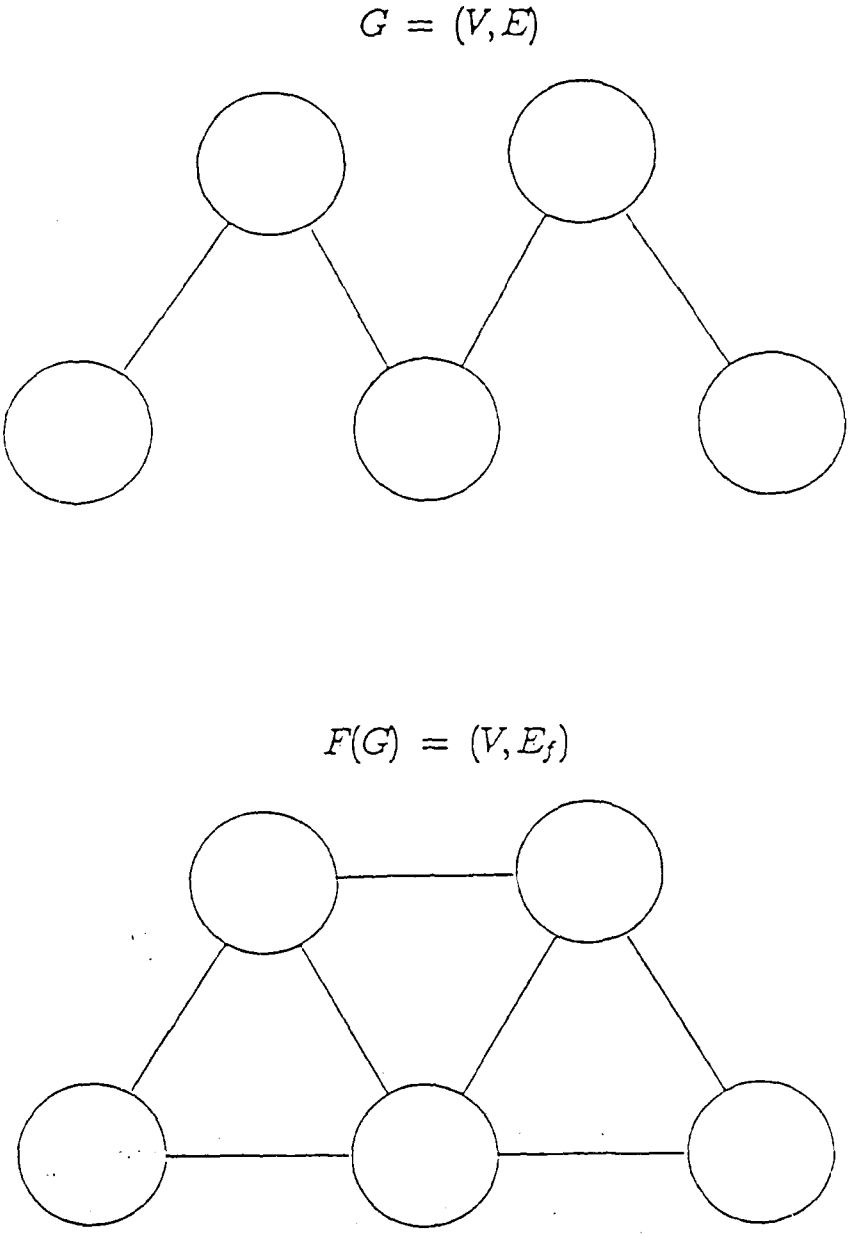


Figure 3-16 Illustration of the tranformation F in Section 3.1.3

$$J(\tilde{x}) = 10x_1^2 + 10x_2^2 + 10x_3^2 + 10x_4^2 + x_1x_2 + x_1x_3 + x_2x_3 + x_2x_4 + x_3x_4$$

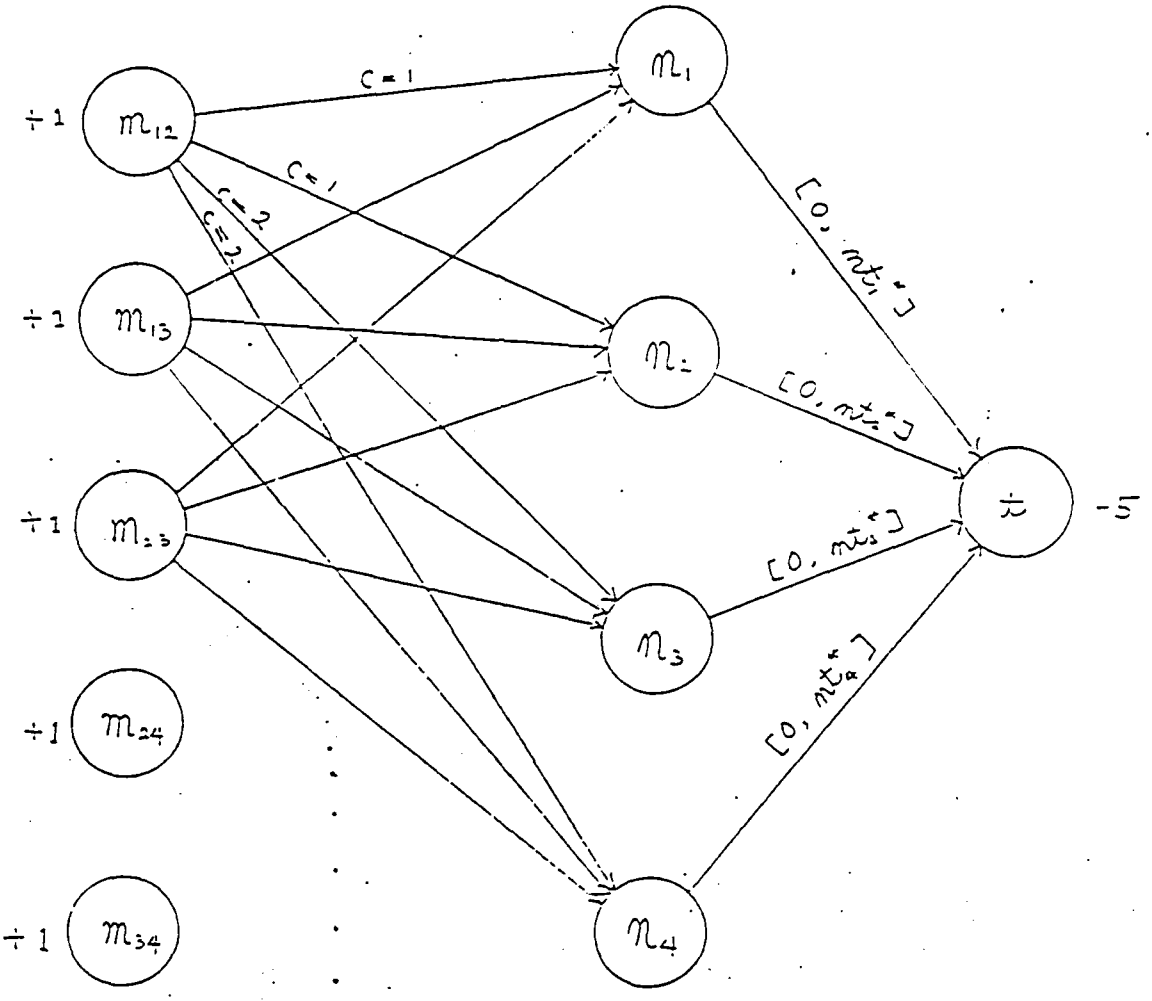


Figure 3-17 Network corresponding to BIL problem

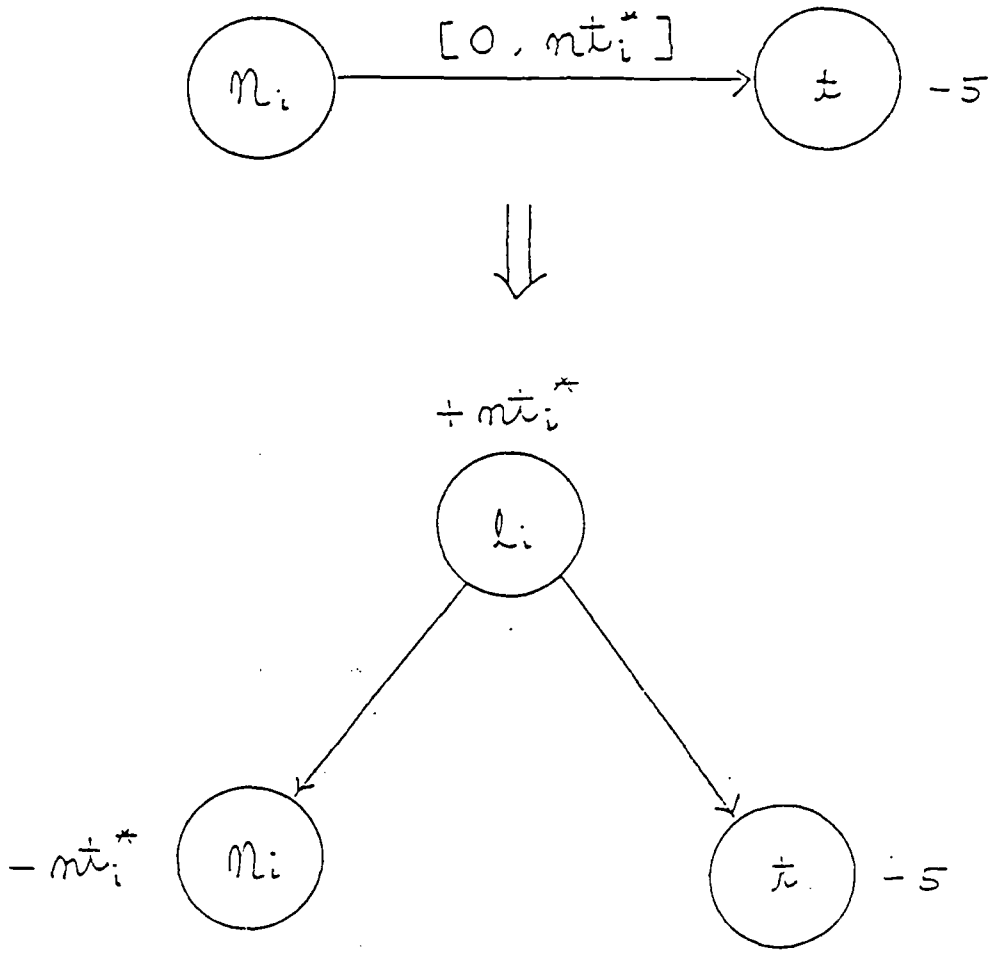


Figure 3-18 Transformation of 'Min-cost flow' to 'Hitchcock'

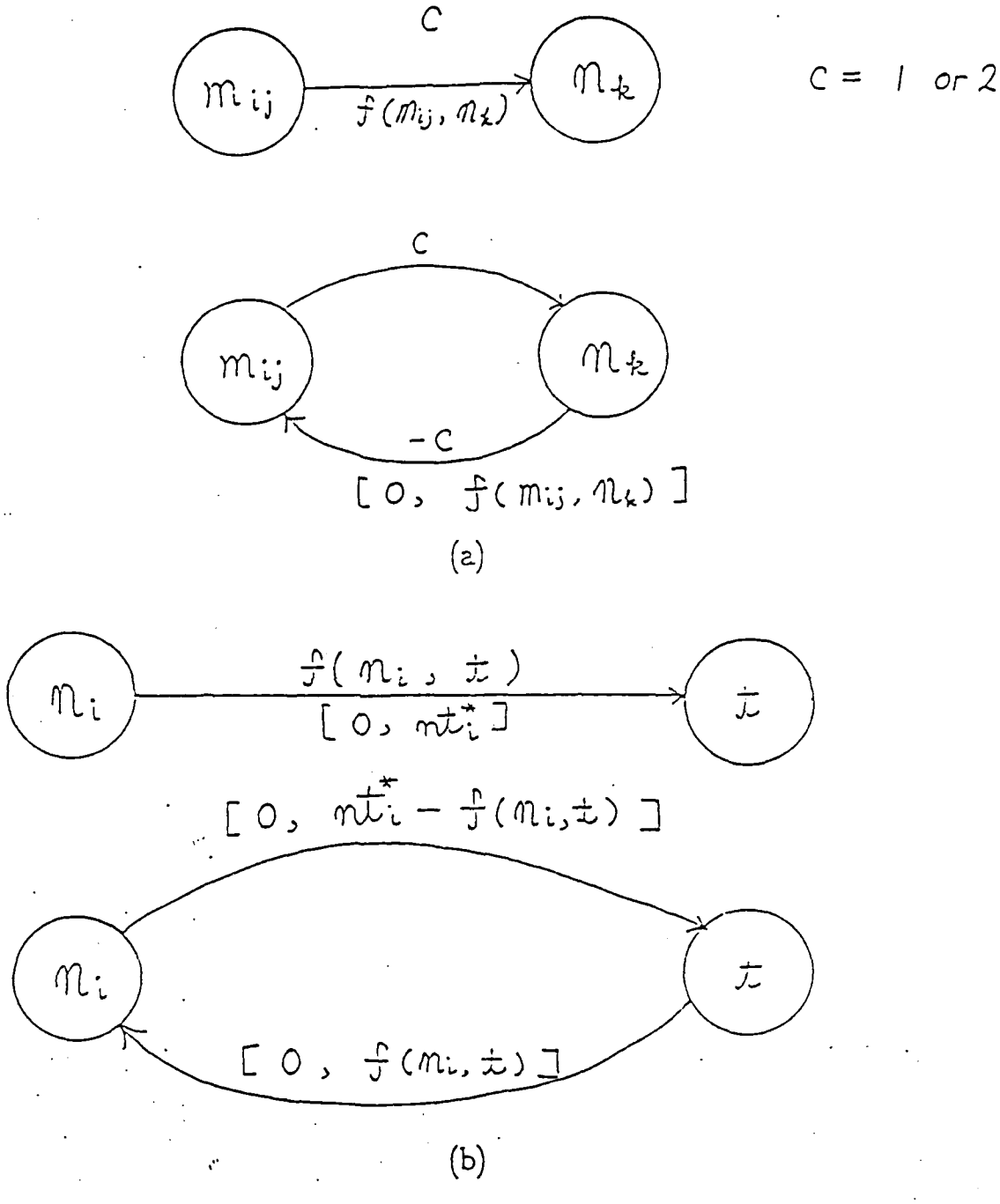


Figure 3-19 Construction of an increment network for a flow, $\tilde{f}$

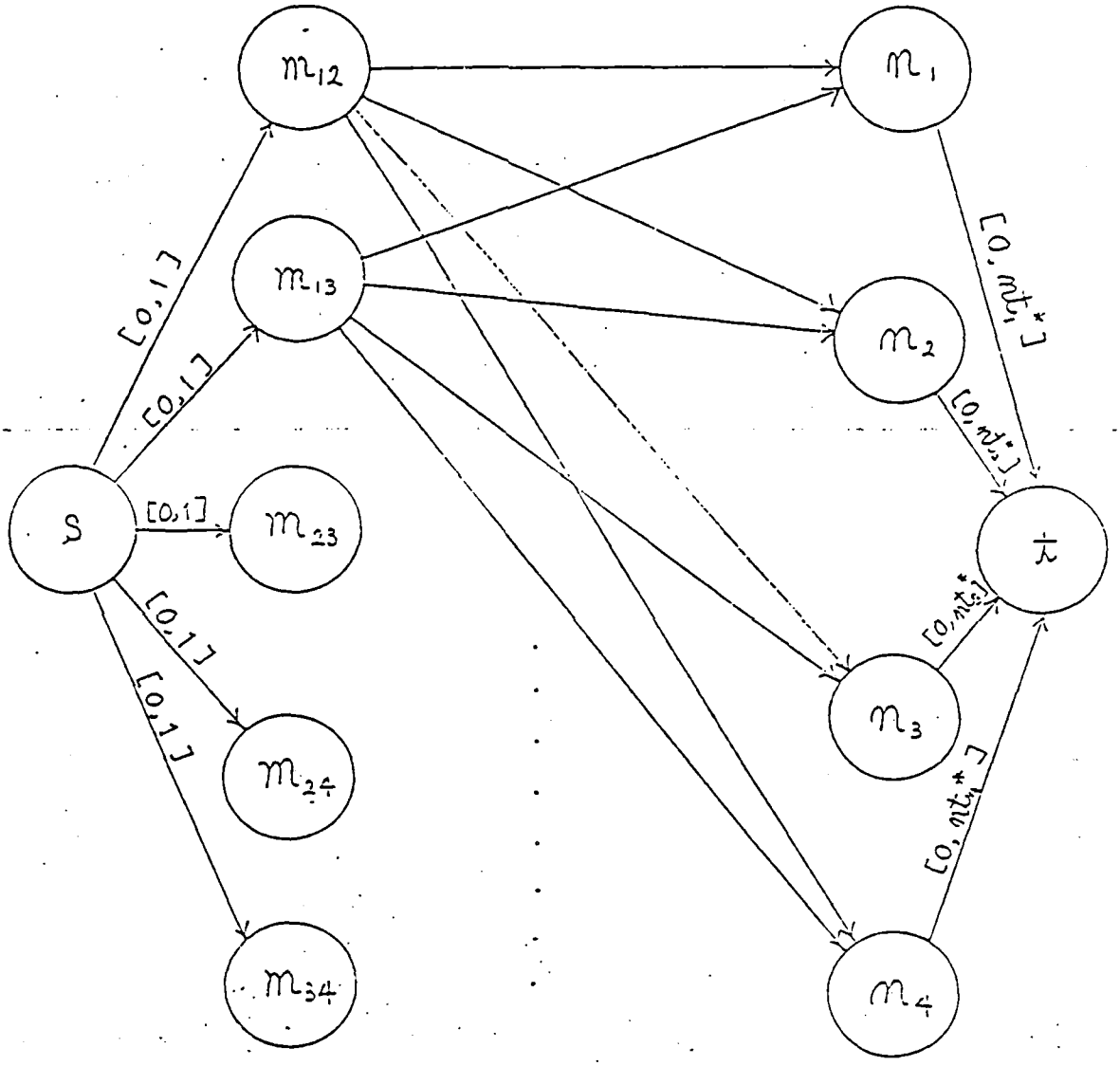


Figure 3-20 Network for obtaining an integral initial flow

Chapter 4. INDIRECT COMMUNICATION

4.1 Introduction

In Chapter 3, we have assumed that if J^i depends upon x_j, there must exist a link between DM_i and DM_j. In this chapter we relax this assumption. We assume that as long as there is an undirected path between DM_i and DM_j, J^i can have x_j as its variable and/or J^j can have x_i as its variable in the decomposition of J. (A path is defined to be a sequence of edges of the form, (DM_1, DM_2), (DM_2, DM_3), (DM_3, DM_4),..., (DM_{l-1}, DM_l). Let us recapitulate the assumptions under which we continue our discussion.

Let $J = x^T Q x$ be the global cost function and G_Q be the graph that represents J.

$$G_Q = (V_Q, E_Q)$$

$$V_Q = \{1, 2, ..., M\}$$

$$E_Q = \{ \text{ unordered } (i,j) |\ x_i x_j \text{ is in } J\}$$

Assumption 1 : If J^i depends upon x_j, there is a path between DM_i and DM_j.

('Indirect Communication' assumption)

Assumption 2 : G_Q is connected, and diagonal elements of Q are non–zero.

Assumption 3 : Each square term x_i^2 is in J^i in the decomposition.

Under these assumptions the amount of communication (TL or RL) and the balance of load ($max_i\ nt(i)$) are to be optimized in the decomposition of J. It has been shown in chapter 2 that if TL is chosen as a measure of the amount of communication, the optimization becomes trivial. In this chapter the optimization of RL and $max_i\ nt(i)$ is discussed.

4.2 Flexible organizational structure

Let us consider the cases where the capacity of each processor is fixed, and the load on the processors is considered balanced as long as load of each processor is within its capacity. The objective, then, is to minimize the total amount of necessary communication.

Minimal Superposed Link

Given $J = x^T Q x$, nt_i^*, find a decomposition that minimizes RL such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

The decomposition of a global cost function can be regarded as an assignment of all the cross terms of the global cost function to processors, or a mapping from the set of cross terms to a set of processors. If a cross term $x_i x_j$ is to be assigned to a processor DM_k, where $k \neq i$ and $k \neq j$, information of the updated value of x_i must be continually sent from DM_i to DM_k, and information of the updated value of x_j must be continually sent from DM_j to DM_k. Also, updated values of partial derivatives of the subcost function J^k must be continually sent from DM_k to both DM_i and DM_j. There must be communication between DM_i and DM_k, and also DM_j and DM_k. Since the organizational structure is flexible, and the 'indirect communication' is allowed, the task allocator has a freedom to choose the communication path between each pair of processors that need to communicate with each other. Let the chosen paths be P_{ik} and P_{jk}, respectively. For this assignment of $x_i x_j$, RL is increased by

$$mt(x_i x_j, DM_k, P_{ik}, P_{jk}) = l(DM_i, DM_k, P_{ik}) + l(DM_j, DM_k, P_{jk})$$

where $l(DM_i, DM_k, P_{ik})$ is the length of the path P_{ik} from DM_i to DM_k, and

$l(DM_j, DM_k, P_{jk})$ is the length of the path P_{jk} from DM_j to DM_k. (mt stands for 'message transmission'.)

If a cross term $x_i x_j$ is to be assigned to DM_i or DM_j, there must be a communication between DM_i and DM_j. Again, the task allocator has a freedom to choose this communication path. Let the chosen paths be P_{ij}. RL for this assignment of $x_i x_j$ is, then, increased by

$$mt(x_i x_j,) = l(DM_i, DM_j, P_{ij})$$

Therefore, given a decomposition (or a mapping from the set of cross terms to a set of processors) and path selection between each pair of processors that need to communicate according to this decomposition,

$$RL = \sum_{(i,j) \in G_Q} mt(x_i x_j, ,)$$

The problem is to find a decomposition and a path selection which minimize this RL.

We can observe that for a cross term $x_i x_j$ assigned to DM_k, $k \neq i$, $k \neq j$, the best path between DM_i and DM_k is a direct path, and so is the path between DM_j and DM_k. Therefore,

$$l(DM_i, DM_k, P^d_{ik}) = 1$$

$$l(DM_j, DM_k, P^d_{jk}) = 1$$

where P^d_{ik} is a direct path from DM_i to DM_k, and P^d_{jk} is a direct path from DM_j to DM_k ($P^d_{ik} = (DM_i, DM_k)$, and $P^d_{jk} = (DM_j, DM_k)$) Therefore,

$$mt(x_i x_j, DM_k, P_{ik}, P_{jk}) = 2$$

For a cross term $x_i x_j$ assigned to DM_i or DM_j, the best path between DM_i and DM_j is again the direct path.

$$mt(x_i x_j, DM_i, P_{ij}^d) = 1$$

and

$$mt(x_i x_j, DM_j, P_{ij}^d) = 1$$

for this direct path $P_{ij}^d = (DM_i, DM_j)$. Therefore, once a decomposition is determined, the selected paths ought to be direct ones. Therefore, the problem of selecting communication paths is embedded in the problem of decomposition, so 'Minimal Superposed Link' can be rewritten as

minimize $\sum_{k \neq i, k \neq j} 2X_{ijk} + \sum_{k=i} X_{ijk} + \sum_{k=j} X_{ijk}$

such that

$\sum_{k=1}^{M} X_{ijk} = 1 \qquad \forall (i,j) \in E_Q$

$\sum_{(i,j) \in E_Q} X_{ijk} \quad \leq nt_k^*.$

$X_{ijk} = 0 \; for \; (i,j) \notin E_Q$

$X_{ijk} \in \{0,1\}$

where

$$E_Q = \{ \text{ unordered pair } (i,j) | \text{ the cross term } x_i x_j \text{ is in } J \}$$

This is exactly the same as linear programming formulation of 'Minimal Superposed Link' problem under 'Direct Communication' assumption (Chapter 3). The conclusion is that 'Minimal Superposed Link' problem under 'Indirect Communication' assumption is exactly equivalent to the one under the 'Direct Communication'

assumption. We do not gain anything by relaxing this constraint, because direct communications are the best in order to have small RL.

Load Balancing with Limited Superposed Link

Given $J = x^T Q x$, RL^*, find a decomposition that minimizes $max_i\ nt(i)$ such that $RL \leq RL^*$.

Notice that direct communications are the best in order to have small RL. Since the constraint is $RL \leq RL^*$, we can find an answer to this problem, again, by solving 'Load Balancing with Limited Superposed Link' under 'Direct Communication' assumption.

4.3 Fixed organizational structure

As discussed in the previous section, decomposing a global cost function is exactly equivalent to assigning all the cross terms to processors. If a task allocator or algorithm wants to assign cross term $x_i x_j$ to a processor DM_k, $k \neq i$, $k \neq j$, he must assure the communication path between DM_i and DM_k and between DM_j and DM_k. Unlike the case of 'flexible organizational structure', choice of this communication path is not completely free. Since the organizational structure is fixed, a communication route between a pair of processors must be chosen from the set of paths between this corresponding pair of nodes in the graph representing the organizational structure. The increment of RL for the assignment of this cross term is again the sum of length of these two routes. (If $x_i x_j$ is assigned to either DM_i

or DM_j, the increment of RL is the length of the chosen route between DM_i and DM_j.) Given any assignment of a cross term, the route that minimizes the increment of RL is the shortest path between two processors that need to communicate. (In case of a flexible organizational structure, the shortest path was always the direct path. In fact, we can see the flexible organizational structure as a special case of fixed organizational structure, which is a complete graph.)

Minimal Message Ambulation

Given a graph G, G_Q and nt_i^*, find a decomposition that minimizes

$$\sum_k \sum_{x_i x_j \text{ is in } J^k} nd(DM_i, DM_k) + nd(DM_j, DM_k)$$

such that

$$nt(i) \leq nt_i^* \qquad i = 1, 2, .., M$$

where $nd(DM_l, DM_m)$ is the distance of the shortest path between DM_l and DM_m assuming every edge has a distance 1. ($nd(DM_l, DM_m) = 0$ if $l = m$) (nd stands for 'nominal distance'.)

(We can cast off the constraint on the capacity of each processor by letting $nt(i) = \infty, \forall i$.)

The idea to solve this problem is, like the idea of Chapter 3, to transform the problem to a network flow problem. A network associated with this problem is created by building one node, m_{ij} associated with each cross term $x_i x_j$, and building one node, n_k associated with each agent of the organization, DM_k. Then, links connecting m's and n's are built with some cost of the link. More specifically, for each link (m_{ij}, n_k), let the cost be the sum of the shortest path between DM_i and DM_k and the shortest path between DM_j and DM_k. The solution of min-cost flow problem on this network gives the solution of 'Minimal Message Ambulation'.

Algorithm 4.1

Phase 1

1. Create $|E_Q|$ nodes corresponding to cross terms of J. Let m_{ij} denote the node corresponding to $x_i x_j$.
2. Create M nodes corresponding to processors. Let n_i denote the node corresponding to DM_i.
3. For each pair of nodes $\{m_{ij}, n_k\}$, make an edge (m_{ij}, n_k)

   All the edges created in Step 3 have infinite capacity.
4. Let the distance of every edge in G be 1.

   Compute the distance of the shortest path between each pair of nodes in G.

   (Let us use $sl(i,j)$ to denote the shortest path between the pair $\{DM_i, DM_j\}$.)
5. For each edge (m_{ij}, n_k),

$$cost = \begin{cases} sl(i,j) & \text{if } i = k \text{ or } j = k \\ sl(i,k) + sl(j,k) & \text{if } i \neq k \text{ and } j \neq k \end{cases}$$

6. Create the sink node t and make an edge from each n_i to t. Each edge (n_i, t) created at Step 5 has capacity nt_i^*

Phase 2

Run 'Primal–Dual algorithm'.

This algorithm can be applied to a more general version of a problem, where communication overhead is defined more generally. Let each link of G have its own cost of delivering message; say, the cost of transmitting through (DM_i, DM_j) is $c(i,j)$. If a cross term $x_i x_j$ is to be assigned to a processor DM_k, where $k \neq i$ and

$k \neq j$, there must be communication between DM_i and DM_k, and also DM_j and DM_k. Let communication paths P_{ik} and P_{jk} be chosen. For this assignment of $x_i x_j$, communication overhead is increased by

$$Cost(x_i x_j, DM_k, P_{ik}, P_{jk}) = \sum_{(p,q) \in P_{ik}} c(p,q) + \sum_{(p',q') \in P_{jk}} c(p',q')$$

If a cross term $x_i x_j$ is to be assigned to DM_i or DM_j, there must be a communication between DM_i and DM_j. Let the chosen paths be P_{ij}. Communication overhead is, then, increased for this assignment of $x_i x_j$ by

$$Cost(x_i x_j, , P_{ij}) = \sum_{(p,q) \in P_{ik}} c(p,q)$$

Therefore, given a decomposition (or a mapping from the set of cross terms to a set of processors) and path selection between each pair of processors that need to communicate according to this decomposition,

$$Communication\ Overhead\ = \sum_{(i,j) \in G_Q} Cost(x_i x_j, DM, P, \ )$$

Algorithm 4.1 can be modified to minimize this *Communication Overhead*. We only modify Step 4 of Algorithm 4.1.

Algorithm 4.2

Phase 1

1. Create $|E_Q|$ nodes corresponding to cross terms of J. Let m_{ij} denote the node corresponding to $x_i x_j$.
2. Create M nodes corresponding to processors. Let n_i denote the node corresponding to DM_i.
3. For each pair of nodes $\{m_{ij}, n_k\}$, make an edge (m_{ij}, n_k).

   All the edges created in Step 3 have infinite capacity.

4. The distance of each edge in G, (DM_i, DM_j) is $c(i,j)$.

   Compute the distance of the shortest path between each pair of nodes in G.

   (Let us use $sl(i,j)$ to denote the shortest path between the pair $\{DM_i, DM_j\}$.)

5. For each edge (m_{ij}, n_k),

$$cost = \begin{cases} sl(i,j) & \text{if } i = k \text{ or } j = k \\ sl(i,k) + sl(j,k) & \text{if } i \neq k \text{ and } j \neq k \end{cases}$$

6. Create the sink node t and make an edge from each n_i to t. Each edge (n_i, t) created at Step 5 has capacity nt_i^*.

Phase 2

Run 'Primal–Dual algorithm'.

We can also consider $max_i nt(i)$ as our objective while setting the amount of communication defined by RL as a constraint.

Load Balancing with Limited Message Ambulation

Given a graph G, G_Q, RL^*, find a decomposition that minimizes $max_i nt(i)$ such that

$$\sum_k \sum_{x_i x_j \text{ in } J^k} nd(DM_i, DM_k) + nd(DM_j, DM_k) \leq RL^*$$

We can apply the same idea that was used in chapter 3 under 'Direct Communication' assumption.

Algorithm 4.3

1. Set $dummy = \lceil \frac{|E_Q|}{M} \rceil$
2. Run Step 1 throug Step 5 of Phase 1 of Algorithm 4.2
3. Create the sink node t and make an arc from each n_i to t. Each arc (n_i, t) created at Step 4 has capacity $dummy$, and cost 0.
4. Do while $dummy \leq |E_Q|$

   Run Phase 2 of Algorithm 4.2

   If $RL \leq RL^*$, $min\ max_i\ nt(i) = dummy$; terminate

   $dummy := dummy + 1$
5. Return "Infeasible"

4.4 Semi–flexible organizational structure

Mapping for Minimal Message Ambulation

Given G_Q, $G = (V, E)$, and nt_i^* for $i = 1, 2, .., M$, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize

$$\sum_k \sum_{x_i x_j \ in \ J^k} nd(\sigma(x_i), \sigma(x_k)) + nd(\sigma(x_j), \sigma(x_k))$$

such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

This problem is very similar to 'Mapping for Minimal Superposed Link' problem. The only difference is that we relax the constraint of 'direct communication'. Let us restate 'Mapping for Minimal Superposed Link' problem.

Mapping for Minimal Superposed Link

Given G_Q, $G = (V, E)$, and nt_i^* for $i = 1, 2, .., M$, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize RL such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$.

'Mapping for Minimal Superposed Link' is basically a combinatorial optimization problem. A feasible domain is a set of pairs, (σ, decomposition) such that:

1. If cross term $x_i x_j$ is assigned to $\sigma(x_k)$, $k \neq i$,$k \neq j$, a link exist in G between $\sigma(x_i)$ and $\sigma(x_k)$ and between $\sigma(x_j)$ and $\sigma(x_k)$.
2. If cross term $x_i x_j$ is assigned to $\sigma(x_i)$ or $\sigma(x_j)$, a link exist in G between $\sigma(x_i)$ and $\sigma(x_j)$.

This feasible domain is defined because we have preassumed that Assumption $1'$: If J^i depends upon x_j, there is a link between DM_i and DM_j.

('Direct Communication' assumption) Assumption 2 : G_Q is connected, and diagonal elements of Q are non–zero. Assumption 3 : Each square term x_i^2 is in J^i in the decomposition.

'Mapping for Minimal Message Ambulation' is formulated from 'Mapping for Minimal Superposed Link' by relaxing Assumption 1. 'Mapping for Minimal Message Ambulation' can be stated as the following:

Problem 4.1

Given G_Q, $G = (V, E)$, and nt_i^* for $i = 1, 2, .., M$, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize RL such that $nt(i) \leq nt_i^*$, $i = 1, 2, ..., M$ under the following assumptions:

Assumption 1 : If J^i depends upon x_j, there is a path between DM_i and DM_j.

('Indirect Communication' assumption) Assumption 2 : G_Q is connected, and diagonal elements of Q are non–zero. Assumption 3 : Each square term x_i^2 is in J^i in the decomposition.

Therefore, the feasible domain of this combinatorial problem is a set of pairs (σ, decomposition) such that:

1. If cross term $x_i x_j$ is assigned to $\sigma(x_k)$, $k \neq i$,$k \neq j$, a path exist in G between $\sigma(x_i)$ and $\sigma(x_k)$ and between $\sigma(x_j)$ and $\sigma(x_k)$.
2. If cross term $x_i x_j$ is assigned to $\sigma(x_i)$ or $\sigma(x_j)$, a path exist in G between $\sigma(x_i)$ and $\sigma(x_j)$.

In a very similar manner as Problem 1 of section 3.2.3 (recognition version of 'Mapping for Minimal Superposed Link'), the recognition version of Problem 4.1 turns out to be NP–complete.

INSTANCE: G_Q, G, nt_i^*, for $i = 1, 2, .., M$, RL^*

PROBLEM : Does there exist a one-to-one mapping σ such that

$RL \leq RL^*$ and such that

$nt(i) \leq nt_i^*$, for $i = 1, 2, .., M$?

Proof

If we restrict this problem by making

$$nt(i) = |E_Q| \qquad \forall i$$

and

$$RL^* = |E_Q|$$

this restricted version is, again, equivalent to 'subgraph isomorphism' problem.

Mapping for Load Balancing with Limited Message Ambulation

Given G_Q, $G = (V, E)$, and RL^*, find a one–to–one mapping,

$$\sigma : \{x_1, x_2, ..., x_M\} \longrightarrow V$$

and a decomposition that minimize $max_i nt(i)$ under the constraint,

$$\sum_k \sum_{x_i x_j \ is \ in \ J^k} nd(\sigma(x_i), \sigma(x_k)) + nd(\sigma(x_j), \sigma(x_k)) \leq RL^*$$

The recognition version of this problem is identical to that of 'Mapping for Minimal Message Ambulation'.

Since these problems are NP–complete, one type of approach to these problem is to consider special cases of these problems, and see if an efficient algorithm can be found for those special cases. Let us consider a special case of 'Mapping for Minimal Message Ambulation', where the graph G is linear.

$$G = (V, E)$$

$$V = \{DM_1, DM_2, ..., DM_M\}$$

$$E = \{(DM_i, DM_{i+1}) | i = 1, 2, .., M-1\}$$

This problem models a situation where the organization has a linear communication network, and the task allocator wants to assign jobs to agents of an organization

with minimal amount of necessary communication. It turns out that even this set of special instances is NP–complete. We can restrict this set of instances further by setting

$$nt_i = |E_Q| \qquad i = 1, 2, ..., M$$

This means that all the processors have enough capacity, so the balance of load needs not be considered. Now let us look at this doubly restricted set of instances. Let σ be an arbitrary one–to–one mapping from V_Q to V. For a cross term $x_i x_j$ in J, a simple path between $\sigma(i)$ and $\sigma(j)$ is unique because G is linear. Therefore, as long as $x_i x_j$ is assigned to an agent on this path, $nd(\sigma(i), \sigma(x_k)) + nd(\sigma(x_j), \sigma(x_k))$ is the length of this path. It is the minimal increase of RL for the assignment of $x_i x_j$ for this mapping, σ. Therefore, the recognition version of the set of instances being discussed is as follows:

Given $G_Q = (V_Q, E_Q)$, $G = (V, E)$ that is linear, and RL^*, does there exist a one–to–one mapping,

$$\sigma : V_Q \longrightarrow V$$

such that

$$\sum_{(i,j) \in E_Q} nd(\sigma(i), \sigma(j))$$

Say $\sigma(i) = DM_{i'}$ and $\sigma(j) = DM_{j'}$. Then,

$$nd(\sigma(i), \sigma(j)) = |i' - j'|$$

Therefore, this formulation is equivalent to 'Optimal Linear Arrangement' problem [9], which is NP–complete.

Chapter 5. SUMMARY AND EXTENSIONS

5.1 Summary

In this thesis, the task allocation scheme for an organization is discussed. The objectives of allocation are reduction of individual load, speedy performance, and organizational security. A decentralized gradient–like algorithm for an additive cost function is used as a mathematical model for the behavior of an organization. In this algorithm, each processor DM_i is responsible for the value of one variable x_i, and each processor DM_i has the subcost function J^i. With each processor updating its own variable and communicating information with other processors, the algorithm achieves the goal of optimizing $J = \sum_i J^i$, which is the sum of subcost functions. Minimizing a global cost function is viewed as as a global task of an organization. Each subcost function J^i is associated with the task of each division of an organization. Therefore, decomposing a global cost function represents allocating a global task to divisions. How to decompose J is a central issue of this thesis.

In a decentralized gradient–like algorithm for an additive cost function, if J^i depends upon x_j, there must be a communication between DM_i and DM_j. The issue of decomposition scheme is discussed under two different assumptions concerning communication method of this type of pair, $\{DM_i, DM_j\}$. The 'Direct Communication' assumption mandates that there must be a direct (bidirectional) communication link between DM_i and DM_j. Decomposition of J under this assumption is discussed is Chapter 3. The 'Indirect communication' assumption allows communication messages between DM_i and DM_j to be relayed by other processors. It is only required that there exist an (undirected) communication path between

DM_i and DM_j. Decomposition of J under this assumption is discussed is Chapter 4. In both chapters the objective of decomposition is to balance the load on each processor, and to keep the amount of communication small.

The results are summarized in summary charts at the end of this chapter.

5.2 Extensions

As mentioned in Section 5.1, if J^i depends upon x_j, there must be a communication capability between DM_i and DM_j. Such communication may take place through a single link or a sequence of links depending upon whether we have the 'Direct Communication' assumption or the 'Indirect Communication' assumption. In this thesis we assumed that every communication link is a bidirectional link. If unidirectional links are used instead of bidirectional links, problems formulated in this thesis are modified. Instead of the undirected graph $G = (V, E)$ used in order to represent the communication structure in this thesis, the directed graph $G_d = (V, E_d)$ must be used. G_d having a directed edge $(DM_i, DM_j) \in E_d$ signifies that the information of x_i can be transmitted from DM_i to DM_j, and the information of $\lambda_i^j = \frac{\partial J^j}{\partial x_i}$ can be transmitted from DM_j to DM_i. 'Direct Communication' assumption must be modified to the following: if J^j depends upon x_i, there must be a directed link (DM_i, DM_j) in G_d. 'Indirect Communication' assumption must be modified as the following: if J^j depends upon x_i, there must be a directed path from DM_i to DM_j in G_d. In this thesis TL and RL have been used as a measure of the amount of communication. RL can be used in this new formulations. However, TL must be modified. TL was defined to be the total number of bidirectional links, required by a decomposition (in case of flexible organization), or built in an organization (in case of fixed or semi–flexible organization). In a new (extended

) formulation, the number of 'unidirectional' links must be counted. We can use TDL (Total number of Directed Links) to denote this new measure of the amount of communication. Suppose J^j depends upon x_i, *and* J^i depends upon x_j in a decomposition. TL is counted as one for the pair $\{DM_i, DM_j\}$, but TDL is counted as two for this pair, because a unidirectional link (DM_i, DM_j) and a unidirectional link (DM_j, DM_i) are both required.

These new formulations are suggested for the future research.

Summary chart for 'Direct Communication'

	TL	*RL*
Fixed Organization	Algorithm 3.2	Algorithm 3.5 Algorithm 3.6
Flexible Organization	Open problem (NP–complete conjectured)	*BIL* Algorithm 3.4
Semi-flexible Organization	Open problem (NP–complete conjectured)	NP–complete

Summary chart for 'Indirect Communication'

	TL	*RL*
Fixed Organization	Trivial	Algorithm 4.1 Algorithm 4.2 Algorithm 4.3
Flexible Organization	Trivial	Equivalent to the case of 'Direct Communication'
Semi-flexible Organization	Trivial	NP–complete

APPENDIX

In this Appendix, we present an example of the graph that does not have an inverse image of the transformation defined in Section 3.1.3.

In Chapter 3, $F(G)$, a transformation of the graph $G = (V, E)$, and set of graphs, Σ were defined as the following:

$$F(G) = (V, E_f)$$

$$E_f = E \cup E_{added}$$

where

$$E_{added} = \{(DM_i, DM_j) \notin E | DM_i \textit{ and } DM_j \textit{ are two hops from each other}\}$$

$$\Sigma = \{\textit{graph } H \mid \exists \textit{ a graph } \Gamma \textit{ such that } F(\Gamma) = H\}$$

Concerning the transformation F, it has the following property:

Lemma A.1

Let $G_1 = (V_1, E_1)$, and $G_2 = (V_2, E_2)$. Let us define $G_1 \cup G_2 \equiv (V_1 \cup V_2, E_1 \cup E_2)$. We say G_1 and G_2 are disjoint if $V_1 \cap V_2 = \emptyset$.

If $G = G_1 \cup G_2$, where G_1 and G_2 are disjoint, $F(G) = F(G_1) \cup F(G_2)$, and $F(G_1)$ and $F(G_2)$ are disjoint.

Proof

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. For any pair $\{v_1 \in V_1, v_2 \in V_2\}$, there is no path between them in G. Suppose $(v_1, v_2) \in E_{added}$. Then, v_1 and v_2 must have been two hops from each other. Contradiction.

Theorem A.1 Necessary condition for being in Σ

If a undirected, connected graph $H = (V_h, E_h)$, $|V| \geq 3$ is in the set Σ, for any edge $(u, v) \in E_h$, there exist a node $k \in V_h$ such that $(u, k) \in E_h$ and $(v, k) \in E_h$.

Proof

Since $H \in \Sigma$, there exist a graph $G = (V, E)$ such that $F(G) = H$.

$$V_h = V$$

$$E_h = E \cup E_{added}$$

If $(u, v) \in E_{added}$, u and v must be two hops from each other in G. Therefore, there exists a node $k \in V_h$ such that $(u, k) \in E_h$ and $(v, k) \in E_h$. Now, let us consider the case where $(u, v) \in E_h$. Since H is connected, from Lemma A.1, G is connected, too. Because $|V| \geq 3$, there must be a node $k \in V$ that is a neighbor of u or v in G. If k is a neighbor of both, done. If k is a neighbor of only one of these two, say, u, (k, v) must be in E_{added}. Q.E.D.

Any graph not satisfying this necessary condition does not belong to the set Σ. The graph in Figure A-1 is an example.

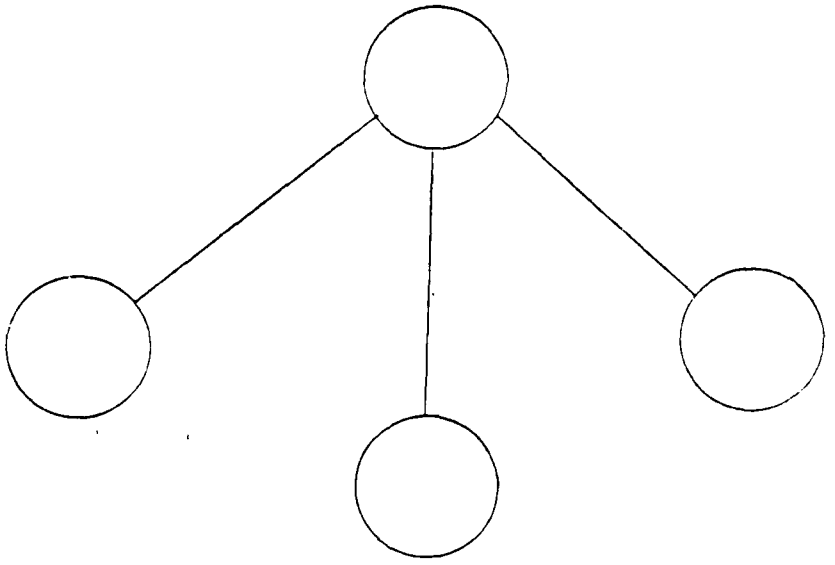


Figure A-1 : Example of a graph that is not in Σ

References

[1] J.N. Tsitsiklis, "Problems in Decentralized Decision Making and Computation", Ph.D. Thesis, MIT, and Laboratory for Information and Decision Systems Report LIDS-TH-1424.

[2] K.L. Boettcher and R.R. Tenney, "On the Analysis and Design of Human Information Processing Organizations", *Proc. 8th MIT/ONR Workshop on C^3 Systems.*

[3] K.L. Boettcher and A.H. Levis, "Modeling the Interacting Decision Maker with Bounded Rationality", *IEEE Trans. on Systems, Man, and Cybernetics*, vol.SCM-12, No.3, pp.334-344, May/June 1982.

[4] D.A. Stabile and A.H. Levis, "The Design of Information Structures: Basic Allocation Strategies for Organizations", *Proc. 6th MIT/ONR Workshop on C^3 Systems.*

[5] Z.J. Wu, P.B. Luh, S.C. Chang, D.A. Castanon, "Optimal Task Allocation for a Team of Two Decision Makers with Three Classes of Impatient Tasks", *Proc. 8th MIT/ONR Workshop on C^3 Systems.*

[6] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Englewood Cliffs,NJ: Prentice-Hall, 1982.

[7] J.N. Clare, "The Role of Teams in the Organization of Work", *Proc. 5th MIT/ONR Workshop on C^3 Systems.*

[8] S.H. Bokhari, "On the Mapping Problem", *IEEE Trans. on Computers*, vol.C-30, No.3, pp.550-557, March 1981.

[9] M.R. Garey and D.S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-completeness.* New York: W.H. Freeman and Company, 1979.

[10] D.P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models.*

Englewood Cliffs, N.J.: Prentice-Hall Inc., 1987.

[11] S.-Y. Lee and J.K. Aggarwal, "A Mapping Strategy for Parallel Processing", *IEEE Trans. on Computers*, vol.C-36, No.4, pp.433-442, April 1987.

END

FILMED

MARCH, 1988

DTIC